METODOLOGIA PARA LA PREVENCION DE INCIDENTES DE SEGURIDAD CRÍTICOS EN APLICACIONES WEB MEDIANTE LA ADOPCION DE ESCENARIOS SEGUROS.

PABLO CESAR VELASCO SOMERA
2046203

UNIVERSIDAD AUTONOMA DE OCCIDENTE
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA INFORMATICA
SANTIAGO DE CALI
2011

METODOLOGIA PARA LA PREVENCION DE INCIDENTES DE SEGURIDAD CRÍTICOS EN APLICACIONES WEB MEDIANTE LA ADOPCION DE ESCENARIOS SEGUROS.

PABLO CESAR VELASCO SOMERA
2046203

Proyecto de Grado para optar el titulo de

Ingeniero Informático

Modalidad: Trabajo de Grado

Director

MIGUEL NAVAS JAIME

INGENIERO DE SISTEMAS

UNIVERSIDAD AUTONOMA DE OCCIDENTE
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA INFORMATICA
SANTIAGO DE CALI
2011

Nota de aceptación:
Aprobado por el Comité de Grado en cumplimiento de los requisitos exigidos por la Universidad Autónoma de Occidente para optar al título de Ingeniero Informático

_____
Firma del presidente del Jurado

_____
Jurado

_____
Jurado

Santiago de Cali, 22 de Febrero de 2011

TABLA DE CONTENIDO

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Los siguientes anexos, se encuentran en formato digital, disponibles en el CD.

INTRODUCCIÓN

La idea de desarrollar este proyecto surge como necesidad de generar una alternativa para la prevención de incidentes de seguridad que se adapte a las necesidades de la compañía, que se adapte al rol del arquitecto de software y al ciclo de vida del desarrollo del software.

Actualmente existen metodologías para la prevención de vulnerabilidades pero la gran mayoría se encuentran enfocadas a los desarrolladores de software, no se encuentran enfocadas al diseño de arquitecturas de seguridad en aplicaciones web sino a la corrección de los incidentes una vez se presenten. Por lo tanto este proyecto se enfoca en las principales recomendaciones de arquitecturas de seguridad para aplicaciones web que han sido generadas a través del tiempo en diferentes compañías y por notables estudios desarrollados por universidades en todo el mundo.

Adicionalmente, para cumplir con el objetivo principal de este proyecto, se realizaron revisiones de los más recientes estudios sobre vulnerabilidades en aplicaciones web. También se investigaron las diferentes metodologías existentes para mostrar la importancia de la seguridad al arquitecto de software. Se involucra al arquitecto de software en el desarrollo de laboratorios controlados para que logre comprender más fácilmente la explotación y aprovechamiento de las vulnerabilidades y finalmente se recolectan aquellas buenas prácticas y conclusiones de los laboratorios que servirán como punto de partida para el desarrollo de la metodología.

Finalmente, se espera que los resultados de este proyecto, sirvan de punto de partida para la prevención de incidentes de seguridad tomando a las vulnerabilidades existentes como estudio para su mitigación desde la fase de diseño del software.

1. TITULO

METODOLOGIA PARA LA PREVENCION DE INCIDENTES DE SEGURIDAD CRITICOS EN APLICACIONES WEB MEDIANTE LA ADOPCION DE ESCENARIOS SEGUROS.

2. PARTICIPANTES

2.1 ESTUDIANTE

Nombre y Apellidos	código	Programa	Modalidad	Email
Pablo Cesar Velasco Somera	2046203	Ingeniería Informática	Trabajo de grado	nupablovel@hotmail.com

2.2 DIRECTOR ACADEMICO DEL PROYECTO

Nombres y Apellidos	Titulo
Miguel José Navas Jaime	Ingeniero de Sistemas

3. DEFINICION DEL PROBLEMA

El propósito de la investigación, es generar una serie de medidas metodológicas para la prevención de incidentes críticos de seguridad que puedan sufrir las aplicaciones web, mediante la generación de escenarios seguros que se consideran dentro de áreas especificas y esenciales para garantizar la seguridad de una aplicación web. Antes de realizar la investigación es necesario cuestionar lo siguiente, ¿Cómo pensar en una aplicación web que logre minimizar las vulnerabilidades críticas definidas en la actualidad?

Debemos considerar entonces los factores determinantes y causantes de los actuales problemas de seguridad existentes en las aplicaciones web, la toma de decisiones y políticas para el aseguramiento de los datos y la información generada por las mismas, una mala administración, prácticas indebidas en la codificación de aplicaciones, carencia de una metodología para planificar una arquitectura durante su fase de diseño y desarrollo influencian en la generación de futuros agujeros de seguridad que podrían ser muy bien aprovechados por usuarios mal intencionados o podrían presentarse en procesos normales de operación, causando muchas veces acceso a información confidencial o control total sobre un sistema de información.

Los arquitectos de aplicaciones web, muchas veces no poseen una metodología definida y orientada de acuerdo a su rol para incluir la seguridad dentro del ciclo de vida de una aplicación web. Gracias a ello, no se cubren totalmente objetivos y puntos neurálgicos para la seguridad, porque se incurre en diseños y especificaciones de la aplicación no orientados hacia la seguridad, por otra parte se busca satisfacer solo requerimientos funcionales de la aplicación trayendo como consecuencia algunos riesgos para las operaciones normales de las organizaciones y su información que con tanto recelo buscan proteger. ¿Qué pasaría si un usuario mal intencionado lograra burlar la aplicación que controla calificaciones en una universidad?. Quizás y pensando en esto, los costos adicionales que genera la solución a problemas de seguridad, podrían disminuir si se incluye a la seguridad como parte del ciclo de vida del software.

Además de los costos adicionales se debe considerar el impacto que tendrá sobre el negocio, sobre los usuarios que utilizan constantemente la aplicación o los problemas sociales que podría desencadenar. ¿Qué pasaría si las calificaciones históricas son borradas?, es entonces cuando la percepción del cliente o usuario de la aplicación cambia y el costo a sufrir por parte de la organización seria un desmejoramiento en la imagen de su producto o servicio.

4. JUSTIFICACION

Este proyecto es importante ya que contribuirá a que los arquitectos adopten un enfoque organizado en el mejoramiento de seguridad en sus aplicaciones web desde que se piensa en el diseño y arquitectura de la aplicación. También contribuirá a generar políticas para el desarrollo seguro de aplicaciones web debido a que la metodología pretende presentar de manera organizada las mejores prácticas para la generación de escenarios menos inseguros basándose en la arquitectura como pilar fundamental.

Se debe considerar la documentación de una metodología adaptada al rol del arquitecto de software de aplicaciones web, dispuesta a ser retroalimentada por las experiencias, mejores prácticas del entorno laboral y productivo con el interés de las personas que quieran adoptarla y contribuir a su mejoramiento.

Teniendo en cuenta que la información en estos momentos es considerada como el activo más importante de las organizaciones. La metodología pretende mitigar las vulnerabilidades que causan incidentes de seguridad con el fin de minimizar los riesgos a los que se ve expuesta la información al momento de pensar en una aplicación web.

La seguridad informática[1] es un campo muy grande que implica muchos esfuerzos e investigación. Actualmente no se ha dado solución a los problemas de seguridad en aplicaciones web adecuando escenarios únicos.

La metodología propone una guía o procedimiento que de ser adecuado a las necesidades y que puede adaptarse a los proyectos de desarrollo de software de cada organización. Se destaca que la metodología puede servir como guía para proyectos que impliquen pensar en mejoramientos y soluciones de seguridad a la construcción de aplicaciones web.

La metodología brinda un ambiente de confianza más elevado para los responsables de asegurar la aplicación desde que se inicia un proyecto de software. La metodología podría disminuir los costos de las organizaciones al adoptarse durante el desarrollo de la misma ya que cuando se mitigan las vulnerabilidades de manera correctiva, es más costoso para la organización

[1] Seguridad Informática, Wikipedia [en línea], 24 de mayo del 2010 [Consultado el 11 de Abril del 2010]. Disponible en internet: http://es.wikipedia.org/wiki/Seguridad_inform%C3%A1tica

trabajar adicionalmente en parches de seguridad.

5. ANTECEDENTES

J.D. Meier, Alex Mackman, Michael Dunner, Srinath Vasireddy, Ray Escamilla y Anandha Murukan generaron una guía para el diseño y el desarrollo de patrones de seguridad en aplicaciones web que hacen uso de tecnologías Microsoft[2]. La guía de arquitectura y diseño de seguridad muestra de manera clara y teórica todos los elementos a considerar que contribuyen al desarrollo de un entorno web seguro.

Esta guía ha sido publicada en internet y puede ser consultada por cualquier persona. La guía hace énfasis en el diseño de arquitectura de la aplicación, donde se informa al lector sobre las mejores recomendaciones de arquitectura de aplicación a tener en cuenta para la seguridad en:

- La validación.
- La autenticación.
- La autorización.
- La configuración.
- Manejo de datos sensitivos.
- Criptografía.
- Manipulación de parámetros.
- Control de excepciones.
- Auditoría.

La guía goza hoy de muy buena aceptación dentro de la comunidad Microsoft ya que se ha convertido rápidamente en un punto de referencia cuando se piensa en diseño y programación segura sobre aplicaciones web bajo tecnología Microsoft.

Actualmente a nivel local, empresas de la región han implementado metodologías para la prevención de las vulnerabilidades tomando en cuenta la metodología de pruebas de OWASP, cuando desean proteger el dinero a través del comercio electrónico.

La mitigación y prevención de vulnerabilidades ha sido aplicada al comercio electrónico en el trabajo de grado: “Definición, Diagnostico y Adecuación de

[2] MEIER, J.D, et al. Architecture and Design Review for Security: Authentication. [en línea]. Microsoft, 2006. [Consultado el 23 de mayo de 2010]. Disponible en: http://msdn.microsoft.com/en-us/library/ff648650.aspx#c05618429_007

Políticas de Seguridad Lógica para la Página Web Comercial de Almacenes la 14 S.A Basado en los Requisitos Incluidos en la Norma PCI DSS", realizado por los estudiantes Leidy Johanna Prado Duque y Jorge Mario Serrate Valencia, egresados de Ingeniería Informática de la Universidad Autónoma de Occidente[3] de Colombia. El objetivo del proyecto es Definir, diagnosticar y adecuar políticas de seguridad lógica para la página web comercial de Almacenes la 14 S.A. basado en los requisitos incluidos en la norma PCI DSS y las recomendaciones de la documentación de OWASP.

Durante el desarrollo del proyecto se definen los requerimientos de la normatividad PCI DSS y las recomendaciones establecidas por las mejores prácticas de programación web generadas por OWASP. Diagnostican las políticas de seguridad lógica de la 14 S.A y su aplicación dentro del sitio web. Después se evalúan si las políticas de seguridad lógica de la 14 S.A cumplen con la normatividad PCI DSS. Posteriormente adecuan las nuevas políticas de seguridad lógica que se ajustan a la normatividad PCI DSS y diseñan un plan de desarrollo para la implementación de modificaciones en la aplicación web.

[3] PRADO DUQUE, Leidy Johanna., SERRATE VALENCIA, Jorge Mario. Definición, Diagnostico y Adecuación de Políticas de Seguridad Lógica para la Página Web Comercial de Almacenes la 14 S.A basado en los Requisitos Incluidos en la Norma PCI. Trabajo de grado para obtener el título de Ingeniero Informático. Santiago de Cali: Universidad Autónoma de Occidente. Facultad de Ingeniería. Departamento de Ciencias de la Información, 2008. 118 p.

6. MARCO TEORICO

6.1 INTRODUCCION A LAS APLICACIONES WEB

Una aplicación Web[4] es un software que realiza una actividad específica a través de Intranet o Internet. La aplicación es utilizada a través de una interfaz gráfica que facilita al usuario el acceso al sistema, para poder mostrar el contenido e información de la aplicación se utiliza un navegador Web.

Generalmente las arquitecturas de las aplicaciones Web se constituyen en tres capas:

- La primera capa corresponde a la presentación, su función es proveer una interfaz entre el usuario y la aplicación, esta capa es accedida a través de los navegadores de internet conocidos (Internet Explorer, Mozilla Firefox, Google Chrome, Opera, entre otros).

- La segunda capa corresponde a la lógica del negocio, su función principal es definir todas las operaciones de la aplicación y ejecutarlas de acuerdo a la tecnología elegida. Generalmente utilizando lenguajes de desarrollo web (Html, php, jsp, .NET, CGI, entre otros).

- La tercera capa corresponde al almacenamiento, es allí donde se guardará la información generada por la aplicación.

[4]Aplicaciones Web [en línea]. Wikipedia. [Consultado el 17 de Abril del 2010]: Disponible en : http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web

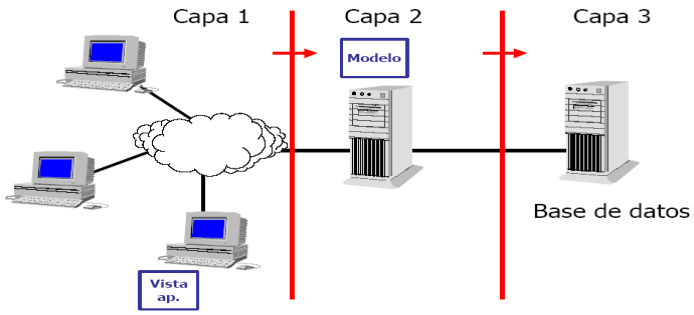
Figura 1. Aplicación web para la administración de clientes

Fuente: Autor

Las aplicaciones Web son fundamentales para mejorar la agilidad de las operaciones que se realizan en las organizaciones, poseen muchas ventajas con respecto a otras soluciones orientadas a la conectividad, alguna de estas son:

- No presentan problemas de compatibilidad ya que pueden ejecutarse en cualquier navegador.
- No ocupan espacio en el disco duro.
- Son multiplataforma.
- Son portables porque no son ejecutadas en la computadora, son ejecutadas en el servidor y accesibles a través del navegador.

Figura 2. Ejemplo de arquitectura de tres capas



Fuente: SANCHEZ, Carlos. Proyecto ONess: Aplicaciones en capas. [En línea]. 2004 [Consultado el 26 de mayo de 2010]. Disponible en: http://oness.sourceforge.net/proyecto/html/images/three_layers.gif

Sin embargo algunas de las aplicaciones son desarrolladas sin tener en cuenta aspectos esenciales, que permiten salvaguardar los pilares fundamentales de la seguridad informática (integridad, disponibilidad, confidencialidad, no repudio entre otros). Actualmente con el crecimiento de las organizaciones en internet, se corren muchos riesgos de ataque a sus aplicaciones Web; los más frecuentes son inyecciones SQL o XSS (Cross Site Scripting) por mencionar solo algunos.

XSS es una vulnerabilidad generada en las aplicaciones Web que permite la ejecución de código en el navegador. Esta ejecución de código se hace embebiendo un script (Java Script o VBScript) en un campo cualquiera de la aplicación (formulario de búsqueda, registro entre otros). Al ser procesada la petición en el formulario se ejecuta el script. (Ver Figura 3)

Un ataque XSS bien elaborado podría ejecutar instrucciones más complejas que permitiría robar cuentas, realizar engaños o estafas a los usuarios que frecuentan la aplicación.

Por este motivo se han desarrollado metodologías para probar la seguridad en aplicaciones web, el proyecto OWASP[5] es una comunidad orientada a los profesionales de la informática, la comunidad busca que los involucrados en el

[5]Acrónimo de Open Web Application Security Project, Proyecto de Seguridad de Aplicaciones Abiertas. Es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro.

proceso de desarrollo de software puedan detectar fallos de seguridad en las aplicaciones web que implementen.

Figura 3. Ejemplo de vulnerabilidad xss.

Fuente: Xss en paypal [En línea], [Consultado el 11 de Abril de 2010]. Disponible en Internet: http://regmedia.co.uk/2009/02/10/paypal_xss_bug.jpg

Metodologías como la guía de pruebas OWASP son empleadas para la realización de pruebas de seguridad en aplicaciones web, la adopción de la metodología contribuye a generar recomendaciones, análisis profundos de la aplicación y riesgos que pueden presentarse dentro de una organización; sin embargo, la adopción de este tipo de soluciones es correctiva. Se propone investigar las vulnerabilidades críticas basándose en estudios realizados por organizaciones de consultoría informática y generar una metodología que permita la creación de escenarios seguros en las aplicaciones web utilizando la prevención como eje principal.

6.2 CONCEPTOS GENERALES

Dentro de los conceptos generales se pretende delimitar la investigación de acuerdo a temas de estudio que se abordaran en la metodología de desarrollo, se mostraran los principales conceptos de interés que serán abordados con mayor profundidad en el proceso de investigación.

6.2.1 Vulnerabilidad. Una vulnerabilidad es el conjunto de factores que posibilitan la aparición de una amenaza. Por lo general las vulnerabilidades aparecen sobre los procesos que manejan activos (información, dinero entre otros) los cuales estarán expuestos a una amenaza (robo del dinero o información).

6.2.2 Amenaza. La amenaza es un evento que puede generar un incidente dentro de una organización según su severidad y posibilidad de ocurrencia. Las amenazas pueden generar daños o pérdidas (tangibles o intangibles) a los activos.

6.2.3 Riesgo. Se considera al riesgo como la proximidad de que una amenaza o un evento no deseado contra un activo en particular ocurra.

6.2.4 Impacto. El impacto tiene como función medir una consecuencia (sea positiva o negativa) cuando se materializa un incidente de seguridad. El impacto pude ser medido cualitativamente asignando palabras que determinan su medición (Bajo, Medio, Alto).

6.2.5 Ataque. Es un evento hostil que puede resultar exitoso o no y que atenta contra el normal funcionamiento de un sistema de información.

6.2.6 Test o prueba de penetración. Un test de penetración consiste en probar diversas técnicas sobre las tecnologías existentes para encontrar falencias o agujeros de seguridad en un sistema de información. Estas técnicas son pequeñas metodologías que permiten abordar y explotar las vulnerabilidades.

Las pruebas de penetración o de intrusión abordan todo el análisis de políticas que hacen parte de la aplicación y un estudio de la misma, se componen de una lista de pruebas que serán abordadas en su fase de ejecución. Luego de la fase de ejecución, posteriormente se genera un reporte de análisis de las vulnerabilidades encontradas y opcionalmente se brinda una solución. Generalmente las soluciones

se traducen en creación de nuevas políticas o reestructuración de los procesos existentes dentro del software incluido en las pruebas.

Actualmente existen metodologías definidas para las pruebas de seguridad en el software, particularmente hacia las aplicaciones web. Una de las metodologías más aceptadas y de libre distribución es OWASP, se han realizado además guías muy extensas basadas en el diseño de aplicaciones web seguras, generadas por Microsoft.

6.2.7 Escenario. Se define como escenario a un conjunto de medidas ampliamente aceptadas que permiten delimitar un comportamiento esperado encaminado a resolver un problema.

6.3 FUNDACION OWASP

La fundación OWASP ofrece un portal con amplia variedad de documentos para las personas involucradas en el área de la seguridad informática. En ella se generan documentos de manera colaborativa con aportes de los usuarios que pertenecen a la comunidad, actualmente posee una amplia popularidad en temas relacionados con la seguridad en aplicaciones web y comercio electrónico. La fundación es reconocida a nivel mundial y goza de amplia aceptación en diversas organizaciones internacionales que han adoptado sus metodologías.

6.3.1 Open Web Application Security Project. El proyecto es dedicado a combatir las causas que generan las vulnerabilidades en las aplicaciones web. OWASP es financiado por muchas empresas de carácter tecnológico alrededor del mundo que junto a personas particulares entusiastas de la seguridad del software, se convierte en una de las mayores comunidades dedicadas a generar soluciones metodológicas y proyectos destinados a mitigar las vulnerabilidades que presentan las aplicaciones web.

Al ser una comunidad abierta, OWASP no es partidaria de ninguna tecnología. Todo el contenido y las aplicaciones desarrolladas bajo el proyecto OWASP pueden ser aplicados en cualquier entorno y plataforma.

6.3.2 Guía de pruebas OWASP. Amplio documento que expone un compendio de vulnerabilidades existentes en aplicaciones web el cual posee una guía de aprovechamiento de vulnerabilidades que se rige mediante una metodología para el análisis de las aplicaciones web. Esta metodología sirve como referencia al

momento de estudiar técnicas existentes al momento de verificar la seguridad de las aplicaciones web, el análisis de vulnerabilidades se realiza mediante la experimentación y uso de algunas herramientas que han sido desarrolladas por la comunidad.

6.3.2.1 ¿Qué es la comprobación?. Durante el ciclo de vida del desarrollo de una aplicación web, los procedimientos del software deben de ser probados para cumplir con los requerimientos definidos inicialmente, se describe a la comprobación como:

- Poner a prueba o probar.
- Pasar una prueba.
- Ser asignado un estado o evaluación basado en pruebas[6].

La comprobación es una secuencia de actividades que permiten comparar el estado actual del software con el deseado o ideal. Cuando se presenta una inconsistencia es cuando podemos diagnosticar entonces un problema lógico o de diseño en la aplicación que se está evaluando.

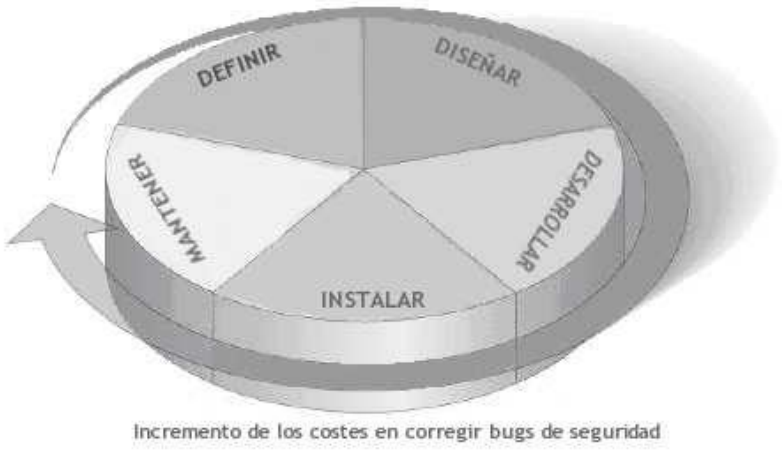
6.3.2.2 ¿Por qué comprobar?. El documento ha sido diseñado para ayudar a las organizaciones a entender la realización de un programa de pruebas, ayudar a identificar los pasos necesarios para construir y operar las pruebas de seguridad sobre sus aplicaciones web. Se pretende suministrar una vista lo más amplia posible de los elementos necesarios requeridos para realizar un programa de seguridad de aplicación web exhaustivo[7].

6.3.2.3 ¿Cuándo comprobar?. Algunas organizaciones no comprueban el software desde sus inicios, solamente mejoran la seguridad de las aplicaciones basándose en una medida correctiva, esta es una práctica muy costosa ya que a medida que se avanza durante el ciclo de vida de un proyecto de software, el costo de corregir fallos o bugs de seguridad se hace mas grande. (Ver Figura 4)

[6] Guía de pruebas OWASP V3.0. [en línea]. 2008 , [Consultado el 26 de Marzo del 2010] Disponible en internet: http://www.owasp.org/images/8/80/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf

[7] Ibíd., Disponible en internet: http://www.owasp.org/images/8/80/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf

Figura 4. Incremento de los costos en corregir fallos de seguridad



Fuente: Guía de pruebas OWASP Versión 3.0, ¿Cuándo comprobar?, [en línea], Marzo de 2008. [Consultado el 26 de Marzo del 2010]. Disponible en: http://www.owasp.org/images/8/80/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf

Las compañías deberían inspeccionar su ciclo de desarrollo de software para asegurar que la seguridad sea una parte integral del proceso de desarrollo, deberían incluir pruebas de seguridad una vez finalizadas sus aplicaciones para corroborar que la seguridad esta adecuadamente cubierta y los controles son efectivos a través del proceso de desarrollo[8].

6.3.2.4 ¿Qué debe comprobarse?. Comprobar el software implica involucrar personas, procesos y tecnología. Se debe evaluar a las personas para comprobar que estas tienen la suficiente concientización y educación para definir un buen software. Se deben comprobar los procesos para poder garantizar que existan políticas y estándares que estén siendo aplicados de la manera debida por las personas y se debe comprobar tecnología para poder garantizar que los procesos sean bien definidos durante su implementación.

6.4 OWASP TOP 10[9]

Documento donde se exponen las 10 vulnerabilidades mas criticas que afectan las aplicaciones web, este documento es actualizado cada 3 años, reúne

[8] Ibíd., Disponible en internet: http://www.owasp.org/images/8/80/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf

[9] OWASP Top 10. [en línea]. Owasp Foundation. Abril del 2010, [Consultado el 26 de Marzo del 2010] Disponible en internet: http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf

información sobre las tendencias de los atacantes y nuevas vulnerabilidades encontradas en las aplicaciones web. A continuación se listaran las vulnerabilidades a las que el documento hace referencia en su orden de importancia en el año 2010[10].

- Injection
- Cross Site Scripting (XSS)
- Broken Authentication and Session Management
- Insecure Direct Object References
- Cross Site Request Forgery (CSRF)
- Security Misconfiguration
- Insecure Cryptographic Storage
- Failure to Restrict URL Access
- Insufficient Transport Layer Protection
- Unvalidated Redirects and Forwards

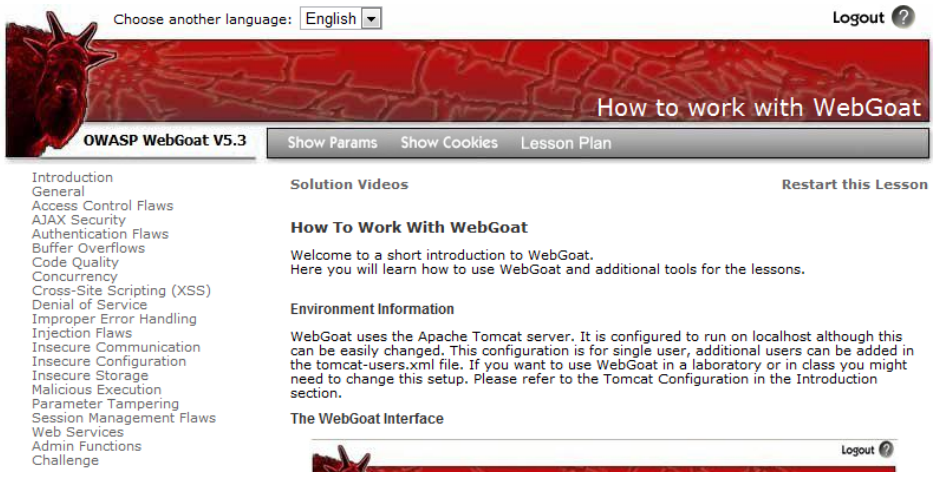
Las vulnerabilidades listadas serán explicadas en la etapa correspondiente al desarrollo del proyecto.

6.5 WEBGOAT

Webgoat es un gestor metodológico para pruebas de penetración que está diseñado para enseñar de manera organizada como se pueden atacar las aplicaciones web. Provee un entorno realista, un listado de pruebas extenso y clasificado que el usuario puede conocer de una manera dirigida para tomar cada una de las lecciones. Webgoat está escrito en Java y puede ser ejecutado en todos los sistemas operativos que ejecuten la plataforma. (Ver Figura 5)

[10] OWASP Top 10. [en línea].Owasp Foundation, Abril del 2010, [Consultado el 26 de Marzo del 2010] Disponible en internet: http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf

Figura 5. Pantalla inicial de Webgoat



Fuente: Autor

Al lado izquierdo se encontraran todas las categorías que posee Webgoat para realizar las pruebas. (Ver Figura 6)

Figura 6. Categorías de Webgoat

Fuente: Autor

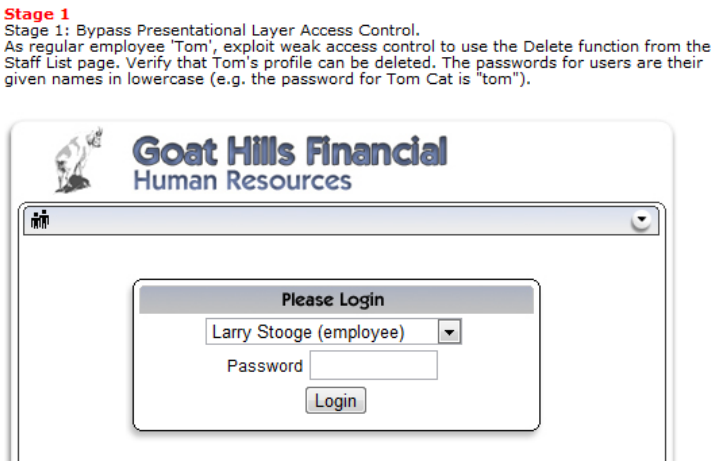
Al seleccionar una categoría, se pueden apreciar las distintas pruebas que contiene. (Ver Figura 7)

Figura 7. Pruebas de la categoría “Access Control Flaws”.

Fuente: Autor

Al seleccionar una prueba, se puede detallar el objetivo. (Ver Figura 8)

Figura 8. Prueba, “Bypass Business Layer Access Control”.



Fuente: Autor.

Una vez seleccionada la prueba, se debe resolver en base a los conocimientos del usuario relacionados con el ataque a las aplicaciones web.

Webgoat es una aplicación web construida para ser vulnerable, está organizada por categorías las cuales pueden ser resueltas de acuerdo a las habilidades de la persona que la utiliza. Es de uso libre y puede ser descargada sin ningún costo[11].

6.6 SOBRE LA ARQUITECTURA DE SOFTWARE

IEEE[12] define a la arquitectura de software como el proceso en el cual se detallan los componentes de un sistema de acuerdo a sus funciones, interacción con el entorno, comunicación y principios que guían su diseño y evolución.

La arquitectura de software se encarga de detallar las funciones de un componente en base a sus requerimientos. Un componente de software es un recurso desarrollado para cumplir con los requerimientos específicos, es considerado como un sistema independiente dentro de un software o aplicativo en general.

6.6.1 Rol del arquitecto de software[13].

En general, el arquitecto de software debe cumplir algunas de las siguientes funciones relacionadas con el tratamiento y diseño del software:

- Liderar el proceso de arquitectura.
- Generar entregables: documentos de descripción de los componentes de arquitectura.
- Generar modelos de arquitectura.
- Visualizar el comportamiento de un sistema.
- Responsable de integrar los requerimientos no funcionales en el sistema.

Las buenas prácticas generadas serán consideras como requerimientos de seguridad que deberá cubrir el diseño de arquitectura de la aplicación.

[11] Descarga de Webgoat: http://code.google.com/p/webgoat/downloads/list

[12] 1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems. [en línea]. IEEE Computer Society. [Consultado el 24 de Julio de 2010]. Disponible en: http://standards.ieee.org/findstds/standard/1471-2000.html

[13] KAPPER. El rol de los Arquitectos de Software. [en línea]. [Consultado el 26 de Abril de 2010]. Disponible en: http://www.epidataconsulting.com/tikiwiki/tiki-read_article.php?articleId=7

7. OBJETIVOS

7.1 OBJETIVO GENERAL

Desarrollar una metodología para la prevención de incidentes de seguridad en las aplicaciones web mediante la adopción de escenarios seguros.

7.2 OBJETIVOS ESPECIFICOS

- Investigar sobre las vulnerabilidades mas criticas en las aplicaciones web.
- Desarrollar los laboratorios sobre las vulnerabilidades más críticas utilizando el gestor metodológico Webgoat.
- Investigar sobre las mejores prácticas utilizadas para la prevención de las vulnerabilidades mas criticas.
- Desarrollar la metodología propuesta para el caso de estudio planteado, mediante las conclusiones, observaciones de los laboratorios, las mejores prácticas y recomendaciones investigadas.

8. METODOLOGIA

La metodología propuesta para la realización de este proyecto, consistió en definir cada una de las actividades para cumplir con cada uno de los objetivos específicos que fueron definidos en el proyecto. Se propuso el desarrollo de las siguientes actividades:

- Investigar sobre las vulnerabilidades mas criticas en las aplicaciones web.

  - Consultar el top 25 de SANS[14].
  - Consultar top 10 de OWASP.
  - Consultar los problemas de seguridad existentes en las aplicaciones web según Microsoft.
  - Definición de las vulnerabilidades más comunes en aplicaciones web.

- Desarrollar los laboratorios sobre las vulnerabilidades más críticas utilizando Webgoat y la metodología OWASP.

  - Listado de pruebas según las vulnerabilidades más críticas.
  - Documentos y conclusiones de cada prueba realizada.

- Investigar sobre las mejores prácticas para la prevención de las vulnerabilidades.

  - Consultar las mejores prácticas para la prevención de vulnerabilidades en las aplicaciones web según el observatorio CWE.
  - Consultar las mejores prácticas según OWASP.
  - Revisar la literatura existente sobre mejores prácticas para la prevención de vulnerabilidades en aplicaciones web.

- Desarrollar el documento de la metodología propuesta para el caso de estudio planteado, mediante las conclusiones, observaciones de los laboratorios, las mejores prácticas y recomendaciones investigadas.

  - Introducción
  - Identificación.
  - Categorización.
  - Definición.

[14] Pagina web: http://www.sans.org/top25-software-errors/

9. DESARROLLO

9.1 INTRODUCCION A LA METODOLOGIA

La metodología descrita a continuación se encargará de recolectar algunas de las mejores prácticas que ayudarán al arquitecto de software a generar escenarios seguros que regirán el diseño de la arquitectura en las aplicaciones web. Definirá cada uno de los componentes mediante ideas o diagramas de acuerdo a las funciones principales otorgadas y serán los encargados de cumplir funciones específicas orientadas al mejoramiento de la seguridad.

La metodología está orientada a los arquitectos de software porque que les permitirá tener ideas generales, estructuradas ante el diseño de arquitectura de las aplicaciones web basadas en las buenas prácticas de arquitectura propuestas en la actualidad por importantes comunidades de seguridad informática. La metodología orientara al arquitecto a elaborar los entregables que son necesarios para cumplir con cada una de las actividades y fases. (Ver tabla 1)

Tabla 1. Fases y actividades entregables de la metodología

Fases	Actividades entregables	
Identificación	Listado de pruebas de vulnerabilidades.	
Categorización	Matriz de categorización de vulnerabilidades. Reportes de pruebas de las vulnerabilidades	
Definición	**Gestión de la autenticación**	Definición de los componentes para la gestión de la autenticación.
		Esquema de componentes para la gestión de la autenticación.
	Gestión de la autorización	Definición de los componentes para la gestión de la autorización.
		Esquema de componentes para la gestión de la autorización.
	Gestión de los datos de entrada	Tabla de caracteres especiales.
		Listado de campos de entrada de la aplicación.
		Definición de los componentes para la gestión de datos de entrada.
		Esquema de los componentes para la gestión de datos de entrada.
	Gestión de la	Definición de los componentes para el manejo de errores.

	configuración y manejo de errores	Esquema de componentes para el manejo de errores.
	Gestión del transporte y almacenamiento seguro	Definición de los componentes para el transporte seguro.
		Esquema de componentes para el transporte seguro.
	Gestión de sesiones	Definición de los componentes.
		Esquema de componentes.

Fuente: Autor.

La metodología será un complemento paralelo dentro del ciclo de vida del software, estaría incluida dentro de la etapa de diseño ya que su enfoque está relacionado con el diseño de arquitectura y escenarios de la aplicación. (Ver Figura 9)

Figura 9. Metodología dentro de la fase de diseño.

Fuente: Autor.

9.2 IDENTIFICACION

Para poder conocer cómo actúan los usuarios mal intencionados mientras atacan una aplicación web o saber las posibles causas de los incidentes comunes en las aplicaciones por parte de los usuarios la frecuentan, se deben identificar y conocer las vulnerabilidades que las afectan. El conocimiento de las vulnerabilidades le

permitirá al interesado considerar soluciones que orienten a un diseño de arquitectura de la aplicación web.

Las vulnerabilidades más críticas incluidas en este estudio se definen mediante las observaciones realizadas a clasificaciones en los problemas de las aplicaciones web y del software publicadas por OWASP (OWASP top 10), al escalafón de las 25 vulnerabilidades más comunes en la programación publicado por SANS (SANS top 25) y un seguimiento al observatorio de vulnerabilidades de software Common Weakness Enumeration[15] (CWE). En este último se puede determinar la importancia de cada una de las vulnerabilidades que serán citadas a continuación sin importar su orden.

9.2.1 Reconocimiento de vulnerabilidades criticas

El reconocimiento de vulnerabilidades críticas tiene como objetivo identificar cada uno de los problemas que afectan la seguridad de las aplicaciones web para posteriormente realizar un estudio detallado en los laboratorios que se realizaran mediante el gestor metodológico Webgoat.

9.2.1.1 Inyección de parámetros. La inyección de parámetros consiste en el envío de instrucciones por medio de los formularios de la aplicación que al ser manipulados por el atacante y enviados al motor de la base de datos sin validar por error de la aplicación, podrán ejecutar sentencias o instrucciones no deseadas para las operaciones normales de la aplicación.

Las inyecciones de parámetros pueden desencadenar grandes problemas para las operaciones normales del gestor de la base de datos de la aplicación, pueden desencadenar la perdida de información, negación de acceso a las cuentas de los usuarios o comprometimiento total del servidor de la aplicación. (Ver Figura 10)

[15] Pagina web: http://cwe.mitre.org/

Figura 10. Información confidencial de los usuarios al ejecutar inyecciones de tipo SQL en una aplicación web.

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA		0
101	Joe	Snow	2234200065411	MC		0
102	John	Smith	2435600002222	MC		0
102	John	Smith	4352209902222	AMEX		0
103	Jane	Plane	123456789	MC		0
103	Jane	Plane	333498703333	AMEX		0
10312	Jolly	Hershey	176896789	MC		0
10312	Jolly	Hershey	333300003333	AMEX		0
10323	Grumpy	youaretheweakestlink	673834489	MC		0
10323	Grumpy	youaretheweakestlink	33413003333	AMEX		0
15603	Peter	Sand	123609789	MC		0
15603	Peter	Sand	338893453333	AMEX		0
15613	Joesph	Something	33843453533	AMEX		0

Fuente: Autor.

9.2.1.2 Vulnerabilidades XSS (Cross Site Scripting). Las vulnerabilidades XSS se presentan cuando un usuario mal intencionado se aprovecha de los formularios de entrada de la aplicación para enviar sentencias mal intencionadas, al ser interpretadas por el navegador web de la victima pueden ocasionar robo de información. Los ataques XSS están relacionados directamente con el robo de cuentas de usuarios, robo de información, direccionamientos a sitios mal intencionados y propagación de malware[16]. La clasificación de los ataques XSS más comunes es la siguiente:

- **XSS reflejado.** Los ataques XSS reflejados son interpretados de manera automática por el navegador al ser enviados a la aplicación desde el mismo navegador del usuario.

- **XSS almacenado.** Cuando se puede almacenar información en las aplicaciones, muchos atacantes almacenan sentencias maliciosas, que al ser accedidas por el navegador, son ejecutadas. Un usuario podría ingresar este tipo de sentencias en formularios de registro de cualquier tipo (registros de usuarios, registros de incidentes entre otros), o en un lugar de la aplicación que permita escribir y guardar contenido.

9.2.1.3 Fallas de autenticación y manejo de sesiones. Los problemas de autenticación y el manejo de sesiones de los usuarios pueden ser explotados cuando el atacante aprovecha sus habilidades para vulnerar las aplicaciones, el robo de identificadores de las cuentas de los usuarios, contraseñas adivinables o

[16] Malware, malicious software. El malware es el software construido con fines mal intencionados.

algoritmos mal implementados para la identificación de los mismos son algunos de los problemas que se presentan.

9.2.1.4 Referencia directa e insegura a un objeto. Los problemas de referencia directa a los objetos de la aplicación pueden ser explotados por los usuarios que tienen acceso limitado, cuando se manipulan las direcciones url o direcciones de internet de la aplicación, se puede acceder a contenidos los cuales no están autorizados y que representan un riesgo muy elevado en las operaciones en el mantenimiento de la aplicación ya que dichos recursos pueden ser eliminados o modificados por el atacante.

Un ataque de referencia directa a un objeto se presenta cuando se accede al contenido de los recursos de una aplicación de la siguiente manera:

http://victima/index.asp?item=pagina.html

Un usuario mal intencionado puede modificar la dirección cambiando el recurso, este hace referencia a una carpeta o archivo dentro de la aplicación:

http://victima/index.asp?item=archivo.sensible

Las vulnerabilidades de referencia directa, pueden ocasionar el acceso total del usuario mal intencionado al contenido de todos los archivos y carpetas de la aplicación.

9.2.1.5 Cross Site Request Forguery. Las vulnerabilidades CSRF[17] permiten la manipulación de las solicitudes que son enviadas a la aplicación web, estas ejecutan acciones que no están contempladas por el usuario pero son validas para la aplicación. Generalmente las vulnerabilidades CSRF son explotadas por los atacantes mediante envíos de enlaces con peticiones maliciosamente modificadas, estas son ejecutadas por la victima mediante un acto de ingenuidad. Este tipo de ataques pueden provocar transferencias electrónicas, robo o modificación de sesiones de usuario, desvío de información o direccionamiento a sitios maliciosos. (Ver Figura 11)

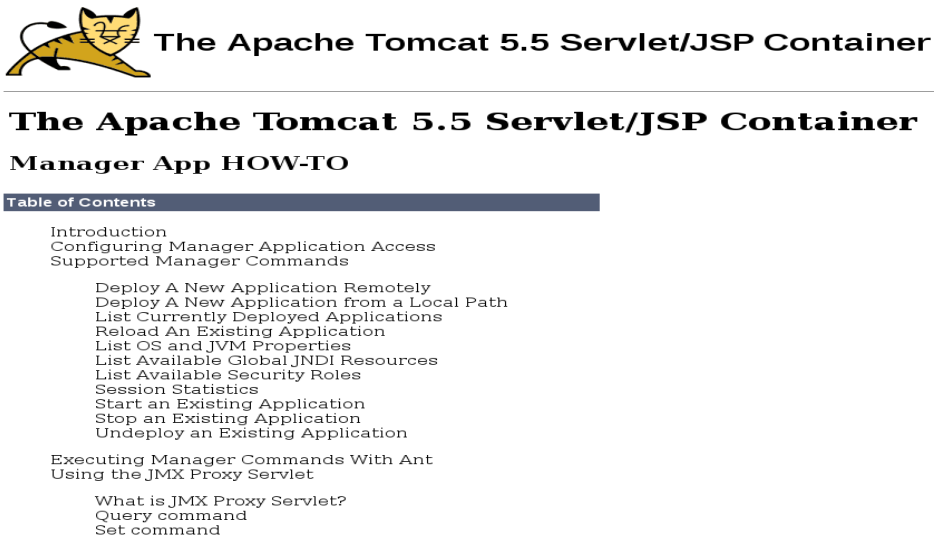
[17] CSRF, o también conocida como Cross Site Request Forgery.

Figura 11. Inclusión de una solicitud maliciosa en un formulario de correo electrónico.

Fuente: Autor.

9.2.1.6 Configuración insegura de la aplicación. De la configuración adecuada de los servicios que componen la aplicación, depende la seguridad de la misma. Cuando las configuraciones de los servicios que soportan la aplicación web no se ajustan a las necesidades o practicas deseables, se pueden producir problemas de permisos, accesos no autorizados a los archivos y carpetas que contienen la aplicación (Ver Figura 12), a contraseñas por defecto facilitan a los atacantes escalar privilegios, volverse dueños de los servicios que aloja el servidor o comprometer el servidor en general.

Figura 12. Acceso total al manual detallado del un servidor web.

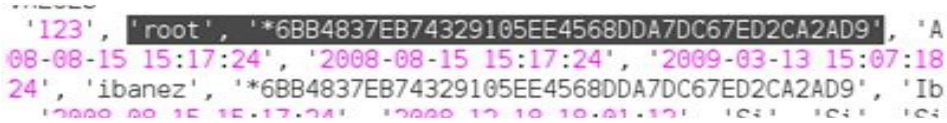


Fuente: Autor.

9.2.1.7 Cifrado y almacenamiento inseguro. Algunas aplicaciones web no protegen la información que generan. La ausencia de un mecanismo que permita proteger la información o la mala implementación del mismo podría dejar al descubierto información que podría ser aprovechada por los atacantes.

Información confidencial como tarjetas de crédito, contraseñas (Ver Figura 13) o información personal de los usuarios son algunos de los ejemplos en los cuales es evidente que la información almacenada por la aplicación no está debidamente protegida.

Figura 13. Almacenamiento inseguro de las cuentas de usuario de una aplicación web protegidas con un mecanismo débil.

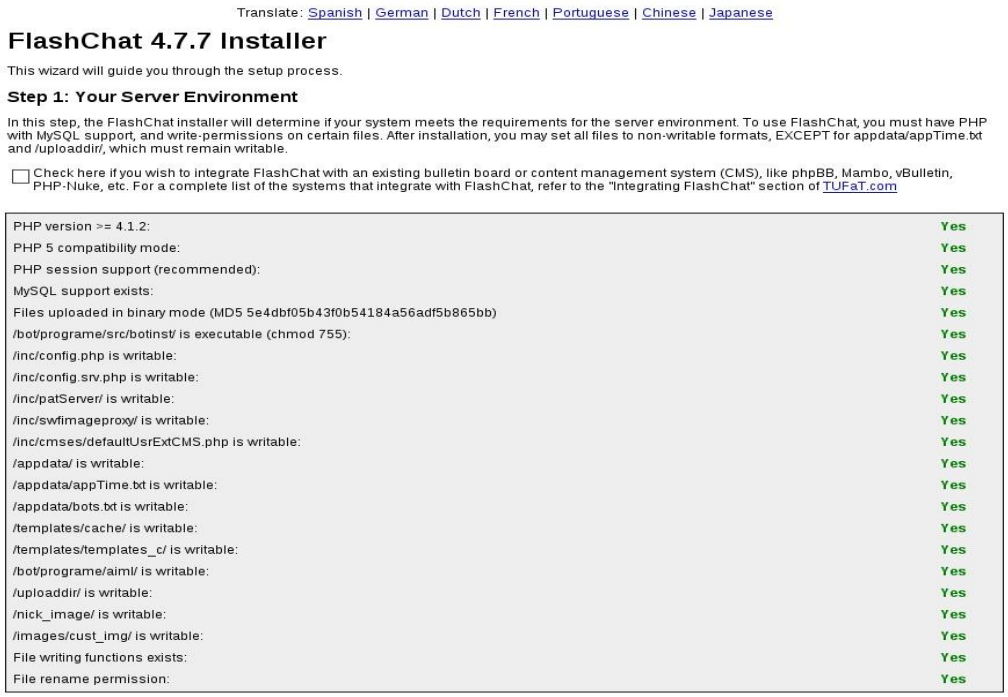


Fuente: Autor.

9.2.1.8 Fallas de restricción de acceso a direcciones de internet . Cualquier usuario que tenga acceso a la aplicación web, puede intentar acceder a páginas

restringidas para los permisos del usuario normal. Muchas veces este tipo de vulnerabilidades se deben a que el acceso a los recursos es mal administrado por la configuración de seguridad establecida en el servidor que hospeda a la aplicación. (Ver Figura 14)

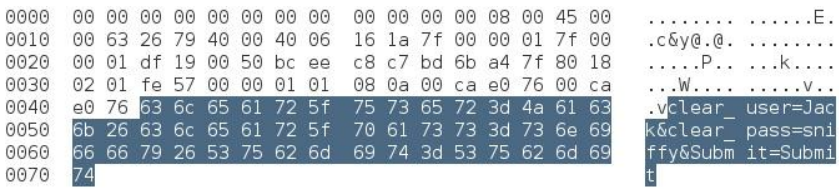
Figura 14. Acceso la pagina de instalación de una aplicación para la comunicación de usuarios.



Fuente: Autor.

9.2.1.9 Protección insuficiente de los datos en la capa de transporte. La protección insuficiente en la capa de transporte se presenta cuando un usuario mal intencionado logra interceptar el trafico que fluye entre la aplicación web y sus usuarios. Una aplicación que no logre proteger toda su información mientras sea transportada, es susceptible a este tipo de vulnerabilidades. Por lo general se utilizan mecanismos que permiten la protección de la información durante su transporte y dificultar la lectura de la información. Un atacante podría utilizar sus conocimientos para robar credenciales de usuarios (Ver Figura 15).

Figura 15. Captura de credenciales de usuario debido a una pobre protección de la capa de transporte de la aplicación.



Fuente: Autor.

9.2.1.10 Direccionamientos y adelantos no autorizados. Algunas aplicaciones web direccionan a sus usuarios a sitios que se encuentran dentro de su confianza pero lo hacen exponiendo totalmente los parámetros de la dirección url. Al notar esta vulnerabilidad, los usuarios mal intencionados pueden modificar la url de la aplicación para redirigir a sus víctimas a sitios maliciosos.

Una aplicación web vulnerable a direccionamientos no autorizados muestra el sitio de destino de la siguiente manera:

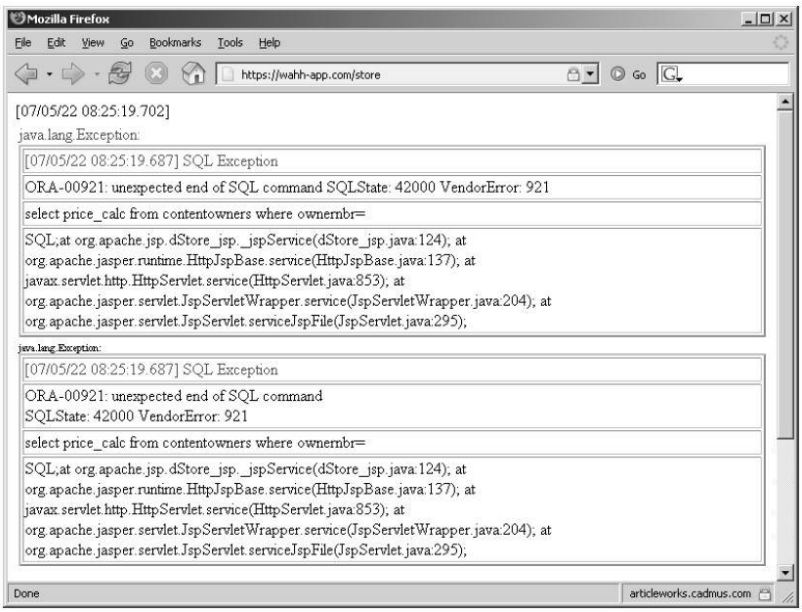
http://www.atacame.com/redirect.asp?url=www.sitiodeconfianza.com

Por lo que un atacante podría modificar la dirección de destino de la siguiente manera:

http://www.atacame.com/redirect.asp?url=www.sitiomalicioso.com

9.2.1.11 Manejo de errores. Algunas aplicaciones muestran sus errores al momento de fallar en determinado procedimiento, estos errores generan mensajes que proveen información detallada sobre la aplicación y son aprovechados por los usuarios mal intencionados para llevar a cabo sus objetivos (Ver Figura 16). La vulnerabilidad es producida ya que los errores no son controlados y capturados durante la fase de construcción de la aplicación.

Figura 16. Mensaje de error provocado por una aplicación web.



Fuente: STUTTARD, Dafydd., PINTO, Marcus. The Web Application hackers handbook. Wiley Publishing, 2008. p. 28.

Uno de los errores comunes presentados de forma indebida a los usuarios tiene que ver con los mensajes mostrados cuando se presentan problemas con el motor de la base de datos de la aplicación. (Ver Figura 17)

Figura 17. Mensaje de error generado por una base de datos

```
Failed to retrieve row with statement - SELECT object_data FROM
deftr.tblobject WHERE object_id = 'FDJE00012' AND project_id = 'FOO' and
1=2--'
```

Fuente: STUTTARD, Dafydd., PINTO, Marcus. The Web Application Hackers handbook. Wiley Publishing, 2008. p. 510.

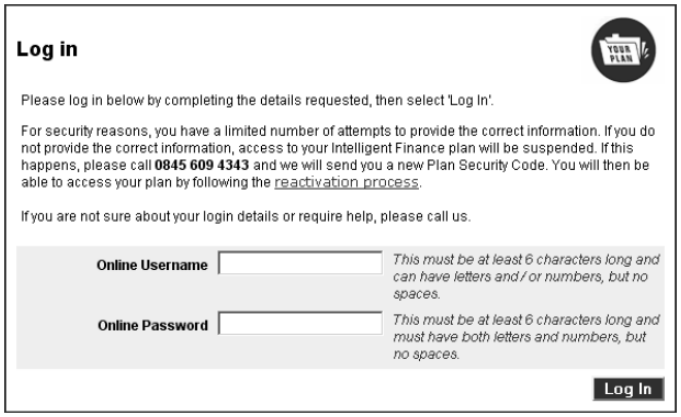
9.2.2 Listado de pruebas de vulnerabilidades. Definidas las vulnerabilidades, es necesario generar un listado de pruebas de penetración, para hacer un estudio más profundo de las vulnerabilidades que se piensan mitigar. Las pruebas del listado fueron realizadas utilizando el gestor metodológico Webgoat. (Ver Anexo A)

9.3 CATEGORIZACION

9.3.1 Definición de categorías. Las categorías permiten agrupar las amenazas más comunes que comprometen la seguridad de las aplicaciones web. A continuación se mostraran las categorías que deben formar parte de la matriz, estarán organizadas por filas y servirán para orientar a los interesados en asegurar las aplicaciones web pensando en diversos escenarios que ayudaran a mitigar las vulnerabilidades conocidas o identificadas como criticas.

9.3.1.1 Gestión de la autenticación. Dentro de la seguridad informática se puede definir a la autenticación como el acto de verificar una identidad ya sea digital o electrónica. Un atacante podría utilizar los problemas de autenticación para poder ingresar a una aplicación web y acceder a información confidencial, obtener el control, escalar privilegios y comprometer todo el sistema de información. La autenticación en aplicaciones web se realiza generalmente utilizando formularios de autenticación. (Ver Figura 18)

Figura 18. Ejemplo de un formulario de autenticación.



Fuente: STUTTARD, Dafydd; PINTO, Marcus. The Web Application Hackers handbook. Wiley Publishing, 2008. p. 17.

9.3.1.2 Gestión de la autorización. La autorización se refleja después de la autenticación, permite conceder acceso a objetos o recursos de un sistema en particular. La autorización y el nivel de acceso que puede lograr un usuario varían de acuerdo a los privilegios que son otorgados mediante permisos, listas de control de acceso u otro mecanismo de autorización.

El manejo de datos sensitivos es el dominio otorgado a los usuarios para el acceso a los recursos de la aplicación web, tales recursos son asignados a los usuarios y accedidos según su dominio. Los problemas en el manejo de datos u objetos sensitivos de la aplicación podrían ocasionar una manipulación indebida de los recursos y acceso a los directorios de la configuración de la misma ocasionando que el usuario mal intencionado pueda recolectar y usar la información suministrada para comprometer la aplicación.

9.3.1.3 Gestión de validación de los datos de entrada. La validación de los datos de entrada consiste en verificar todos los valores que son asignados a los campos que forman parte del suministro a una aplicación web. Los valores que puede tomar un campo en particular pueden variar, los datos de entrada pueden ser caracteres especiales como las comillas simples (') o barras (/), que usadas en los campos de entrada de la aplicación pueden generar sentencias maliciosas que la aplicación envía a la base de datos sin antes ser validada. La finalidad de esta categoría es mostrar todas aquellas medidas y buenas prácticas que permitan filtrar aquellos caracteres o valores que sean determinados como nocivos para la seguridad de la aplicación web.

9.3.1.4 Gestión de la configuración y de errores. La gestión de la configuración consiste en los ajustes de seguridad mínimos que deberían tener los servidores que albergan a la aplicación web. Los errores en la configuración se deben a malos manejos en la administración de los recursos que componen la aplicación o problemas con la administración de los servicios que la soportan. Según Microsoft[18], la gestión de la configuración se refiere a los aspectos operativos de la aplicación, la administración de las bases de datos, de la aplicación y la ejecución de los procesos están incluidos dentro de lo que se debe tener en cuenta al momento de garantizar la seguridad de los servicios que la soportan.

Esta categoría incluye el manejo de errores. El manejo de errores es la capacidad que tiene la aplicación para poder identificar y presentar la información que no puede ser procesada de manera correcta. Muchas veces los errores en las aplicaciones web no son capturados por la lógica de la aplicación o son mostrados al usuario de manera indebida, estos errores pueden mostrar información que podría ser utilizada por un usuario mal intencionado para materializar un ataque contra la aplicación web.

[18] Cheat Sheet: Web application security frame. [En línea]. [Consultado el 22 de Mayo de 2010]. Disponible en Internet: http://msdn.microsoft.com/en-us/library/ff649461.aspx

Según Stuttard y Pinto[19], es muy difícil predecir la manera en la que un usuario mal intencionado se comportara con la aplicación, tales comportamientos podrían desencadenar errores no contemplados cuando una aplicación se encuentra funcionando en producción.

9.3.1.5 Gestión del almacenamiento seguro. Cualquier aplicación web necesita proteger el contenido e información de los medios almacenados, muchas no implementan un mecanismo de cifrado lo suficientemente robusto para proteger los datos e información sensible que generen los procedimientos u operaciones.

La información que es almacenada sin incluir un mecanismo de protección, corre el riesgo de ser leída por usuarios no autorizados con mayor facilidad, es por eso que la finalidad de esta categoría es brindar una serie de recomendaciones e ideas que le permitan al arquitecto pensar en un escenario para gestionar el almacenamiento seguro de la aplicación web.

9.3.1.6 Gestión de sesiones. La gestión de sesiones consiste en manejar la interacción entre el usuario con la aplicación web dentro de un periodo de tiempo determinado. Según OWASP[20] el protocolo HTTP sobre el cual funcionan las aplicaciones web, es un protocolo que no posee estado, es por eso necesario vincular al usuario mediante una solución implementada por terceros. La identificación del un usuario en particular se realiza mediante la implementación de cookies o Identificadores de sesión.

Los problemas en la administración de sesiones pueden ocasionar la pérdida o secuestro de una sesión web, así como el paso de parámetros o procedimientos ejecutados por atacantes. Esta categoría pretende mostrar una serie de alternativas que permitan el aseguramiento del tiempo de los usuarios de la aplicación así como de la seguridad de sus identidades.

9.3.2 Categorizando las vulnerabilidades. Una vez conocidas las categorías de la metodología e identificadas las vulnerabilidades más críticas para la aplicación web a proteger, es necesario construir una matríz que permita clasificarlas. La finalidad de la matriz es organizar todas las vulnerabilidades dentro de las

[19] STUTTARD, Dafydd; PINTO, Marcus. The Web Application hacker's handbook: Core defense mechanisms, Indianapolis: Wiley Publishing, 2008. p. 62.
[20] Owasp Testing guide :Web application penetration testing, Owasp foundation, 2008. p. 146.

categorías mencionadas para tener un control y dominio sobre todas las vulnerabilidades contenidas. Se deben organizar las vulnerabilidades de manera horizontal y las categorías de forma vertical, el numero de vulnerabilidades varia ya que corresponde al arquitecto de software considerar mediante su experiencia o una investigación, las vulnerabilidades que existen en el momento.

Es importante realizar este entregable finalizando esta fase de la metodología ya que permitirá conocer las vulnerabilidades que serán mitigadas en la siguiente fase. A continuación se muestra la matriz que permitirá organizar las vulnerabilidades y categorías.

Tabla 2. Matriz Vulnerabilidades vs Categorías

	V1	**V2**	**.**	**.**	**Vn**
C1					
C2					
.					
.					
Cn					

Fuente: Autor.

En relación con las vulnerabilidades y categorías definidas, se mostrará la matriz correspondiente:

Tabla 3. Matriz Vulnerabilidades vs Categorías respecto a las vulnerabilidades contempladas.

	Inyección de parámetros	**Vulnerabilidades XSS**	**Autenticación y Manejo de Sesiones**	**Referencias directa e insegura a un objeto**	**CSRF – Cross Site Request Forgery**
Autenticación			X		
Autorización				X	
Validación de datos de entrada	X	X			
Manejo de la					

configuración					
Criptografía					
Manejo de sesiones			X		X

	Configuración insegura de la aplicación	**Cifrado y almacenamiento inseguro**	**Fallas de restricción de URL**	**Protección Insuficiente en la capa de transporte**	**Direccionamientos no autorizados**	**Manejo de errores**
Autenticación						
Autorización			X		X	
Validación de datos de entrada						
Manejo de la configuración	X					X
Criptografía		X		X		
Manejo de sesiones						

Fuente: Autor.

9.3.3 Reportes de pruebas de las vulnerabilidades. Los reportes de pruebas de las vulnerabilidades se generan a partir de las pruebas realizadas en el listado (Anexo A). La finalidad de los reportes de pruebas es mostrar al arquitecto la necesidad de la seguridad en la aplicación. En los reportes de pruebas se deben registrar todos los procesos llevados a cabo durante los laboratorios realizados con el gestor metodológico Webgoat. Los reportes deben iniciar con un estudio de la vulnerabilidad a investigar, luego se debe realizar el procedimiento descrito por el gestor metodológico para el desarrollo del desafío o laboratorio y finalmente se deben incluir conclusiones que le permitan al arquitecto pensar en el origen y posible mitigación de las vulnerabilidades. (Ver Anexo B)

9.4 DEFINICION

Una vez clasificadas las vulnerabilidades dentro de la matriz, es necesario considerar algunas recomendaciones de diseño de arquitectura y contramedidas que permitirán al arquitecto proponer un diseño de seguridad detallado de la

aplicación web basado en las buenas prácticas de arquitectura segura que se investigaron.

Se realizó una labor investigativa con diferentes observatorios existentes en comunidades, recolectando las buenas prácticas y metodologías orientadas a la seguridad de las aplicaciones web. Se tomo como marco de referencia inicial Microsoft, la comunidad OWASP, el Observatorio de seguridad informática CWE, artículos de investigación actuales y los reportes generados por las pruebas realizadas durante la fase de identificación.

Al centrar el estudio en la mitigación de aquellos riesgos que afectan las aplicaciones web, es conveniente clasificar aquellas medidas y principios de diseño como las capas que componen a una aplicación web, es recomendable clasificar dentro de cada capa las medidas que deben ser tenidas en cuenta antes de proponer un diseño de arquitectura. Se decide incluir entonces la siguiente clasificación para algunas de las categorías definidas anteriormente.

- Capa de presentación

  - En la capa de presentación proponen medidas de seguridad que deben ser tenidas en cuenta por los arquitectos de la aplicación antes de pensar y proponer un posible diseño de arquitectura. Generalmente el usuario final de la aplicación notará estos cambios en el diseño de la interfaz gráfica de la aplicación.

- Capa de Negocio

  - La medidas de seguridad dentro de la capa de negocio proponen una lógica con la que debería contar una aplicación web cuando se piensa en seguridad, un ejemplo son las validaciones y la comunicación entre los componentes de la aplicación web.

- Capa de datos

  - Las medidas dentro de la capa de datos proponen recomendaciones, principios y mitigaciones que tienen relación con el almacenamiento de la información que genera la aplicación.

La clasificación permitirá organizar de manera más detallada y organizada todas las recomendaciones de diseño y contramedidas propuestas por cada capa dentro de cada categoría propuesta a proteger en la aplicación web.

9.4.1 Gestión de la autenticación. A continuación se mostrarán las medidas de diseño de arquitectura y principios para las diferentes capas de la aplicación web correspondientes a la gestión de la autenticación.

- **Medidas para la capa de presentación**

El proyecto OWASP[21] presenta una seria de medidas y recomendaciones de arquitectura que han sido adecuadas para proteger la capa de presentación de la aplicación cuando se piensa en la seguridad de la autenticación en la aplicación.

- Es recomendable que todas las aplicaciones den la facilidad al usuario de terminar su sesión, esto se hace incluyendo una funcionalidad para el cierre de sesión, que se encuentre siempre presente en toda la aplicación.

- La aplicación web debe proveer al usuario de funcionalidades que le permitan crear una contraseña segura, las contraseñas pueden medirse mediante un indicador de complejidad de contraseñas o adoptando un algoritmo que permita la generación de las mismas.

- No es recomendable la utilización de CAPTCHA[22] como medida para autenticación de un usuario a la aplicación web.

- Se debe evitar por completo utilizar preguntas secretas para el acceso a la aplicación web, ya que podrían ser fácilmente adivinadas por un usuario malintencionado[23].

[21] Guide to authentication: Architectural goals. [En línea]. OWASP, 2005 [Consultado el 23 de Mayo de 2010]. Disponible en Internet: http://www.owasp.org/index.php/Guide_to_Authentication

[22] CAPTCHA: Completely Automated Public Turing Test to tell Computers and Humans Apart, mecanismo que permite la diferenciación entre humanos y maquinas comúnmente utilizado en las aplicaciones web para la prevención de ataques automatizados contra los formularios.

- **Medidas para la capa de Negocio**

- Todas las aplicaciones dentro de la organización deberán compartir un mecanismo autenticación de confianza debidamente probado.

- Todas las funciones y recursos deberán ser protegidos mediante un mecanismo de autenticación común.

- Las aplicaciones deberán funcionar con el más bajo privilegio posible dentro de las funciones correspondientes a los servicios, directorios y bases de datos.

- Las credenciales deberán ser transmitidas solo mediante enlaces cifrados, particularmente las contraseñas ya que son mecanismos débiles de autenticación.

- Es necesario cifrar las credenciales antes de ser almacenadas.

- La aplicación deberá ser capaz de escalar los requerimientos de los algoritmos de cifrado para sus credenciales. Es importante planear la aplicación para una futura transición.

- Las aplicación deberá tener la facilidad para alertar al usuario en cuanto a los intentos de autenticación fallidos y ofrecer una funcionalidad que permita cambiar su contraseña en caso de olvido.

- La aplicación deberá informar al usuario sobre la última fecha de ingreso y proveerá la funcionalidad de reportar accesos fraudulentos.

- La aplicación no deberá suministrar información sobre si un nombre de usuario o una contraseña es incorrecta. Es recomendable utilizar un único mensaje de error que permita cubrir ambos escenarios.

- La aplicación deberá tener funciones administrativas que permitan administrar las cuentas que nunca han sido utilizadas, cuentas inactivas y cuentas que han sido bloqueadas por la aplicación.

- Deberá existir una diferencia lógica entre el bloqueo administrativo y el bloqueo por intento de acceso a la aplicación.

- Es necesario volver a autenticar a los usuarios al momento de ejecutar funcionalidades que sean consideradas como críticas dentro de la aplicación.

- La aplicación no deberá almacenar las credenciales en cookies, cabeceras o campos ocultos[24].Evitar utilizar cuentas y credenciales con altos privilegios al momento de conectarse y comunicarse con los diferentes recursos de la aplicación

- Verificar que se implemente el menor privilegio posible para cada funcionalidad realizada por la aplicación.

- Restringir el acceso al momento de determinar cierta cantidad de intentos fallidos a la aplicación.

- Proveer una funcionalidad que permita al usuario cambiar su contraseña de manera periódica.

- Evitar el uso de autenticaciones débiles[25] ya que implementan mecanismos de cifrado débiles para los actuales sistemas información.

- Analizar como la aplicación distinguirá entre un área pública y un área restringida[26].

➢ **Medidas para la capa de Datos**

- La aplicación no deberá almacenar credenciales en texto plano en la base de datos.

- Planificar dónde y cómo las credenciales del usuario serán almacenadas. Es importante almacenar un hash perteneciente al valor de la contraseña y un salto aleatorio para su verificación.

Una vez se consideren los principios de diseño correspondientes a la autenticación, es necesario realizar un esquema conceptual que le permita facilitar e incluir todas las medidas mencionadas con anterioridad. Es muy importante definir los componentes, mostrar su interacción con el sistema y delegar sus

[24] Guide to authentication: Architectural goals, Op. cit., Disponible en Internet: http://www.owasp.org/index.php/Guide_to_Authentication

[25] STUTTARD, Dafydd; PINTO, Marcus. The Web Application Hackers Handbook: Web application technologies. Indianapolis:Wiley Publishing, 2008. p. 47.

[26] Improving Web Application Security: Architecture and Design Review for Security [En línea]. Microsoft, 2006 [Consultado el 23 de Mayo de 2010]. Disponible en Internet: http://msdn.microsoft.com/en-us/library/ff648650.aspx#c05618429_007

funciones con base en las medidas de las buenas prácticas planteadas.

- **Definición de los componentes**

Para definir los componentes se deben definir funciones de las cuales se encargara cada uno de ellos, por ejemplo una función para el componente administrativo seria “modificación de las cuentas de los usuarios”.

Una vez se determinen los componentes para la seguridad de la aplicación en la categoría a proteger, es necesario centrar la atención en la capa mas critica de la aplicación. Se recomienda centrar la atención en la lógica del negocio de la aplicación porque que en esta se encuentran todas las medidas de protección y codificación segura de la aplicación, además se tendrán en cuenta todas las medidas para la prevención de las vulnerabilidades mencionadas con anterioridad. A continuación se describirá cada una de las funciones para cada componente dentro de la capa de negocio. . (Ver Figura 19)

- **Componente administrativo.** El componente administrativo deberá realizar las siguientes funciones:
  - Controlar el estado de todos los usuarios.
  - Tener un control y categoría sobre las funciones críticas de la aplicación[27].
  - Modificación de las cuentas de los usuarios.
  - Seccionamiento de las áreas de la aplicación.

- **Componente de autenticación.** El componente de autenticación deberá realizar las siguientes funciones:
  - Comprobación de las credenciales de usuario.
  - Comprobación de intentos fallidos.
  - Autenticación en funciones críticas.

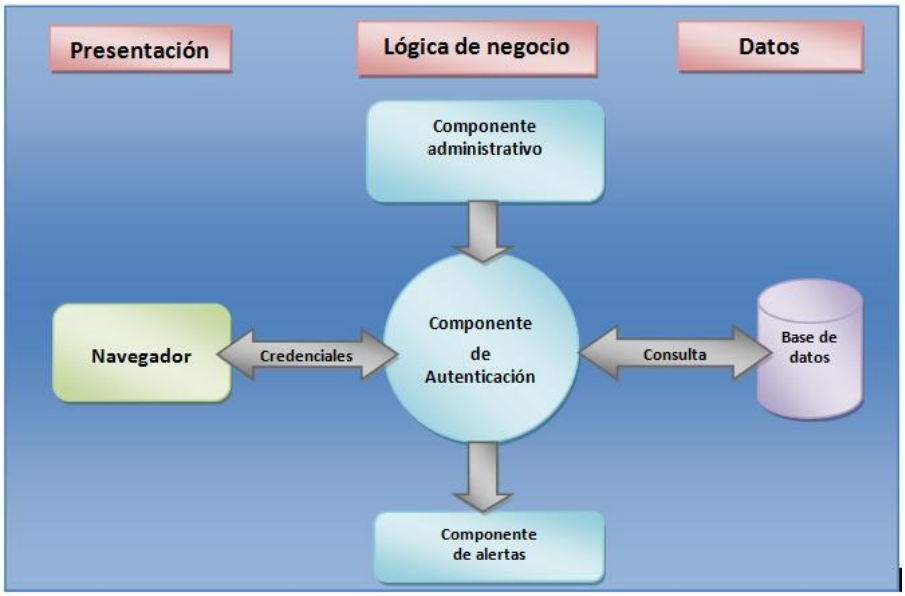
[27] Las funciones críticas son todas aquellas que manipulen activos valiosos para el entorno. Dinero o información privada pueden ser consideradas como activos valiosos.

- **Componente de alertas.** El componente de alertas deberá realizar las siguientes funciones:

  - Alertas de intentos de acceso fallidos.

  - Registro de acceso exitosos.

Figura 19. Esquema conceptual de los componentes para la gestión de la autenticación.



Fuente: Autor

9.4.2 Gestión de la autorización

➢ **Medidas para la capa de presentación**

- Utilizar una página de renuncia de responsabilidad intermedia que proporcione al usuario una advertencia clara de que abandonara la aplicación[28] para dirigirse a un sitio completamente diferente.

[28] CWE-601: URL Redirection to Untrusted Site ('Open Redirect'): Potential mitigations [En línea].Common Weakness Enumeration, 2010 [Consultado el 24 de Mayo de 2010]. Disponible en Internet: http://cwe.mitre.org/data/definitions/601.html

- Implementar un tiempo de espera mucho antes de efectuarse el direccionamiento a un sitio web, o forzar al usuario a hacer clic en el enlace al cual desea seguir.

➢ **Medidas para la capa de Negocio**

- El desarrollo, las pruebas y la puesta en producción de la aplicación deben realizarse con los privilegios más bajos[29].
- Los servidores que alojan las aplicaciones web nunca deberán ser ejecutados como administradores del sistema. Se debe garantizar que las cuentas del sistema tengan privilegios limitados.

- Es recomendable crear cuentas sin privilegios, los privilegios se deben conceder a las cuentas a medida que sean necesarios.

- No utilizar las mismas cuentas de la aplicación como cuentas de los servidores que la administran.

- No base el control de la autorización solamente en los cortafuegos de aplicaciones web (waf[30]). Piense también en otras alternativas que le permitan mitigar las fallas que puedan llegarse a presentar[31].

- Implemente permisos de acceso sobre las carpetas del servidor web, permisos sobre el sistema de archivos del sistema, permisos para la autorización y direccionamientos hacia otra url.

- Si necesita más autorizaciones a determinados niveles de la aplicación, es recomendable crear nuevos roles en vez de conceder permisos a los existentes[32].

La mejor manera de administrar la autorización de los elementos y recursos de una aplicación web es utilizando un índice, un mapa de referencia indirecta u otro método que permita acceder a los recursos de la aplicación mediante un identificador. Esta solución impedirá consultar directamente los recursos de la

[29] Guide to Authorization: Principle of least privilege [En línea]. OWASP, 2009 [Consultado el 25 de Mayo de 2010]. Disponible en Internet: http://www.owasp.org/index.php/Guide_to_Authorization

[30] WAF. Web Aplication Firewall o Cortafuegos de Aplicaciones Web, utilizado para controlar la autorización de los usuarios y recursos de la aplicación web

[31] Architecture and design review for security: Authorization [En línea]. Microsoft, 2006 [Consultado el 25 de Mayo de 2010]. Disponible en Internet: http://msdn.microsoft.com/en-us/library/ff648650.aspx#c05618429_008

[32] Ibíd., Disponible en Internet: http://msdn.microsoft.com/en-us/library/ff648650.aspx#c05618429_008

aplicación.

Según OWASP[33], para evitar problemas de autorización de los recursos de la aplicación es necesario saber que si una referencia a un objeto o recurso de la aplicación web va a ser utilizado, se debe asegurar de que el usuario que la solicita se encuentre autorizado para hacerlo. Se debe evitar exponer los recursos privados a los usuarios, como contraseñas o nombres de archivos sensitivos y verificar la autorización de todos los mismos.

CWE[34] recomienda utilizar un mecanismo que permita implementar un mapa de acceso por referencia a los recursos de la aplicación, utilizar librerías o componentes de autorización como JAAS u OWASP ESAPI Access Control. CWE también hace énfasis en los siguientes aspectos a tener en cuenta para el manejo de la autorización en las aplicaciones web:

- Debe utilizar un mecanismo que le permita autorizar todas las url que realicen peticiones a un recurso en particular.

- Asegúrese que el mecanismo de control de acceso se implemente desde el servidor.

- Los usuarios no deben estar autorizados para acceder a cualquier funcionalidad o información no autorizada simplemente solicitando de manera directa el recurso.

- Divida la aplicación en áreas: área anónima, área normal, área privilegiada y área administrativa.

- Utilice un control de acceso basado en roles, para poder delimitar de manera adecuada las zonas.

Se deben implementar límites al momento de ejecutar los procesos que requieran de acceso a algún recurso ubicado dentro del sistema de archivos de la aplicación web, esto permitirá crear un perímetro de seguridad entre los procesos propios de la aplicación y el sistema operativo. Con la ayuda del perímetro de seguridad se

[33] Insecure direc object reference: Protection [En línea]. Owasp, 2010 [Consultado el 25 de Mayo de 2010]. Disponible en Internet:
http://www.owasp.org/index.php/Top_10_2007-Insecure_Direct_Object_Reference
[34] Improper Access Control (Authorization): Potential mitigations [En línea]. CWE, 2010 [Consultado el 25 de Mayo de 2010]. Disponible en Internet:
http://cwe.mitre.org/data/definitions/285.html

podrán administrar todos los archivos a los cuales podrá acceder la aplicación web así como los comandos que puedan ser ejecutados por la misma.

Cuando se conocen con exactitud todos los objetos, recursos y directorios a los cuales puede acceder la aplicación, es buena idea asignar un valor a cada recurso. Luego se deben rechazar todos aquellos recursos que no se encuentren contemplados. Para llevar a cabo tal labor, puede conocer más sobre librerías encargadas de esta labor como ESAPI Access Reference Map[35].

- **Medidas para la capa de datos**

- Es recomendable utilizar cuentas limitadas en la base de datos que permitan únicamente la ejecución de sentencias permitidas, dichas cuentas no podrán ejecutar sentencias que permitan la modificación del esquema de la base de datos, es importante contemplar que el acceso y operaciones se realicen mediante procedimientos almacenados.

- Examinar como la aplicación autoriza a sus usuarios, también debe examinar como su aplicación autoriza los procesos dentro de la base de datos y como accede a los diferentes niveles y recursos de almacenamiento. Se debe limitar el acceso a la base de datos, tablas, procedimientos y sentencias definiendo muy bien los roles que deberán cumplir los usuarios dentro de la base de datos.

- **Definición de los componentes**

A continuación se describirá cada una de las funciones para cada componente dentro de la capa de negocio para la gestión de la autorización. (Ver Figura 20)

- **Componente de autorización.** El componente de autorización deberá realizar las siguientes funciones:

  - Controlar el nivel de acceso para todos los usuarios.

  - Permitir o denegar el acceso de los usuarios.

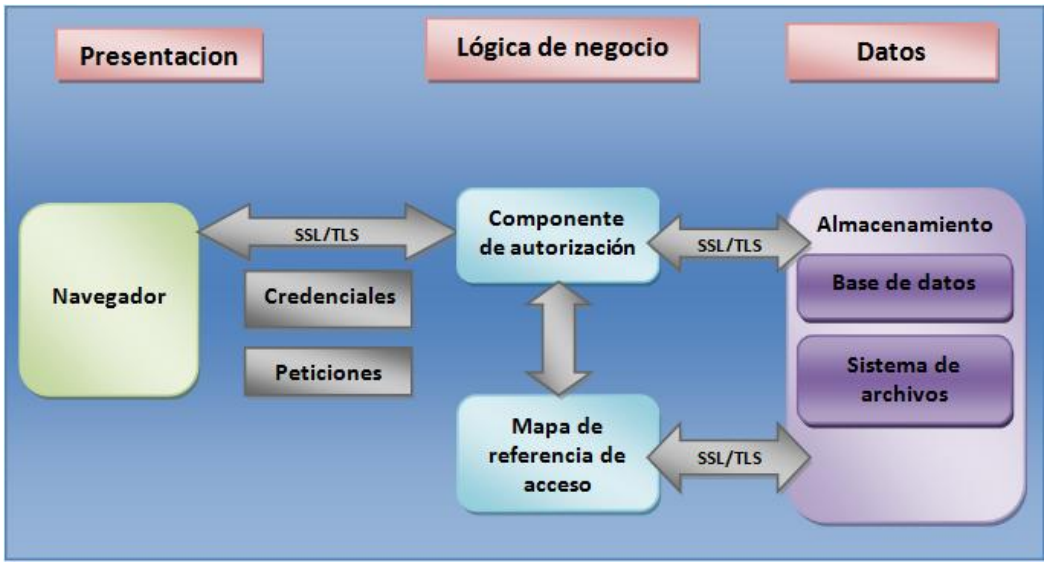
  - Control de los permisos de los usuarios.

[35] Pagina Web: http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/AccessReferenceMap.html

- Control sobre el acceso a los recursos almacenados
- Control sobre el acceso a los recursos url.
- Control sobre el acceso a los directorios.
- Control de roles.

- **Mapa de referencia de acceso.** El mapa de referencia de acceso deberá realizar las siguientes funciones:
  - Control sobre todos los recursos de la aplicación web.
  - Control sobre los recursos almacenados permitidos por la aplicación.
  - Control sobre funcionalidades propias del sistema operativo.

Figura 20. Esquema conceptual de los componentes para la gestión de la autorización.



Fuente: Autor

9.4.3 Gestión de los datos de entrada. Es importante conocer todos los datos que pueden ser perjudiciales para la seguridad de una aplicación web. Los filtros implementados del lado del cliente de la aplicación son muchas veces burlados por los atacantes que implementan herramientas o hacen uso de programas que capturan y manipulan la información mucho antes de ser enviada al servidor. La

validación de datos de entrada provee una protección contra ese tipo de incidentes.

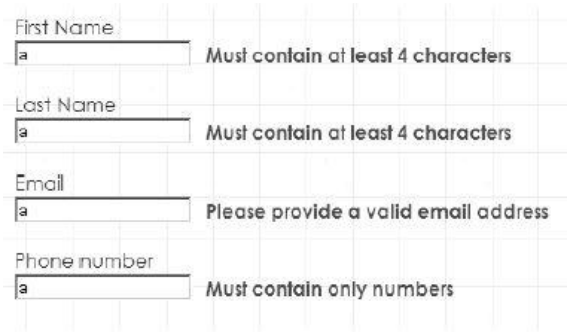
Según Meier[36], es muy importante no confiar toda la validación de los datos de entrada en la capa de presentación, esta tendencia reduce el tiempo de respuesta pero no es confiable cuando se piensa en seguridad ya que dichas validaciones pueden ser burladas con facilidad. Es necesario pensar cómo podemos validar todos aquellos datos que son suministrados a la aplicación. Un buen método se conoce como "permitir sobre denegar" el cual consiste en excluir todos los caracteres contemplados y luego permitir solamente aquellos que la aplicación requiera.

Ya que los problemas de validación de datos de entrada deben ser minimizados desde que se piensa en la aplicación, es necesario citar algunas recomendaciones a tener en cuenta antes de pensar en un escenario para mitigarlos.

- **Medidas para la capa de presentación**

Se deben mostrar al usuario todas las restricciones de los campos pertenecientes a la aplicación web, es importante especificar la longitud, tipo y formato de los campos pertenecientes a la aplicación web. La siguiente imagen nos mostrara la manera más adecuada de presentar información de las restricciones de los campos cuando son presentados al usuario. (Ver Figura 21)

Figura 21. Restricciones de los campos en la capa de presentación



Fuente: STUTTARD, Dafydd; PINTO, Marcus. The Web Application Hackers Handbook. Wiley Publishing, 2008. p. 55.

[36] MEIER, J.D. Web application security enginering: Security engineering baseline activities. En: IEEE Security & Privacy. 2006. p. 16-24.

- **Medidas para la capa lógica de negocio**

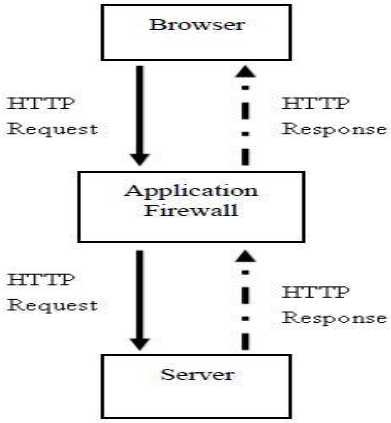
Es recomendable pensar en un componente que permita validar los caracteres que son suministrados a la aplicación. Apache validator[37], ESAPI validator[38] o Java web application security framework[39] son algunos de los componentes que permiten realizar validaciones de entrada y omisión de caracteres especiales.

Muchas aplicaciones web acceden directamente a los comandos que provee el sistema operativo en el cual se encuentran alojadas, esta práctica es muy peligrosa ya que permitirá a un usuario mal intencionado ejecutar comandos[40] desencadenando un posible control sobre el sistema operativo. Es necesario entonces crear implementaciones propias de los comandos que puedan ser usados para los procesos de la aplicación, solamente si son requeridos.

Se puede implementar un cortafuegos para la aplicación web que permita validar y corregir los caracteres de entrada suministrados a la aplicación. El cortafuegos recibe las peticiones del navegador, luego revisa todos los caracteres no deseados de las peticiones comprobando las cabeceras[41] del protocolo http y a través de un corrector suprime todos aquellos caracteres que son considerados como peligrosos para la aplicación. (Ver Figura 22)

[37] Pagina web: http://commons.apache.org/validator/
[38] Pagina web: http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/Validator.html
[39] Pagina web: http://sourceforge.net/projects/hdiv/
[40] Comando: Orden que el usuario proporciona a un sistema informático, dentro del contexto informático es la instrucción proporcionada al sistema operativo
[41] Cabecera: Conjunto de datos que serán procesados y que se sitúa al principio de un bloque de información.

Figura 22. Cortafuegos para la aplicación web



Fuente: TSAI, Dwen-Ren, et al. Optimum Tuning of Defense Settings for Common Attacks on the Web Applications: Defense metodology. En: Security Technology, 2009. 43rd Annual 2009 International Carnahan Conference on. 2009. p. 89-94

Para validar los caracteres que recibe una aplicación, es necesario identificarlos. Es por eso que se recomienda crear una lista de caracteres especiales prohibidos por la aplicación. Esta lista se implementara con el componente de validación del cortafuego para su posterior exclusión de la petición. (Ver Figura 23)

Figura 23. Lista de caracteres especiales

Table 1 Special Symbols

|	Vertical bar or Pipe sign
&	Ampersand
*	Asterisk
;	Semicolon
^	Caret
$	Dollar sign
%	Percent or Percent Sign
@	At sign
'	Single quote
\'	Single quote with escape backslash
"	Double Quote
\"	Double quote with escape backslash
\	Backslash
<	Less than
>	Greater than
(	Open parenthesis
)	Close parenthesis
+	Plus or Plus Sign
,	Comma
CR	Carriage Return
LF	Line Feed

Fuente: TSAI, Dwen-Ren, et al. Optimum Tuning of Defense Settings for Common Attacks on the Web Applications: Defense metodology. En: Security Technology, 2009. 43rd Annual 2009 International Carnahan Conference on. 2009. p. 89-94

Puede realizar un listado de todos los campos correspondientes a la aplicación. El listado de los campos de la aplicación debe especificar el tipo de dato que recibirá el campo, la longitud y el rango de valores que podría llegar a tomar. La siguiente tabla muestra como se deben organizar los campos de la mejor manera.

Tabla 4. Tabla de especificaciones de los campos de la aplicación.

	Tipo de dato	**Longitud (Caracteres)**	**Rango**
Campo 1	Carácter	1	a-z/A-Z/0-9
Campo 2	Cadena	10	a-z/A-Z/0-9
Campo 3	Numérico	8	0-9
Campo 4	Numérico	7	0-9

Fuente: Autor.

- **Tipo de dato:** El tipo de dato puede ser un carácter, una cadena o un número.
- **Longitud:** El tamaño máximo que puede alcanzar el valor del campo, medido en caracteres.

- **Rango:** los valores que puede llegar a tomar el campo, los rangos están determinados por las consideraciones de las personas que desarrollaran la aplicación. En este caso se consideran los rangos del alfabeto en minúsculas (a-z), alfabeto en mayúsculas (A-Z) y números (0-9).

Es necesario tener en cuenta las validaciones para los campos ocultos de la aplicación, muchas veces estos campos no son mostrados por la aplicación ya que llevan un control sobre las características y preferencias de los usuarios. Se debe tener en cuenta incluir en las validaciones, la omisión de código html[42] que pueda ser enviado por un usuario mal intencionado para afectar la aplicación web.

Para prevenir vulnerabilidades XSS[43] es necesario incluir dentro del cortafuegos web, un modulo que impida que ese tipo de vulnerabilidades se presenten. El modulo permitirá deshabilitar todas aquellas etiquetas y palabras propias de los lenguajes tipo script[44] (Java script o VBScript) en los formularios y puntos de entrada de la aplicación. Algunas librerías permiten desarrollar implementaciones adecuadas para evitar esta clase de malas prácticas. Microsoft anti- XSS library[45], Owasp ESAPI encoder module[46] y Apache Wicket[47] pueden ser un gran punto de partida para pensar en un componente contra los ataques XSS.

- **Medidas para la capa de datos**

Utilizar consultas con parámetros permitirá forzar al desarrollador de la aplicación a definir todo el código de la consulta en un procedimiento almacenado, a continuación deberá pasar cada parámetro a la consulta que los procesará. Este estilo de codificación le permite a la base de datos de la aplicación distinguir todos los datos que recibe antes de poder procesarlos.

[42] HTML: HiperText Markup Language, Lenguaje de de marcado de hipertexto. Es lenguaje más popular para la construcción de páginas web.
[43] XSS: Cross Site Scripting, vulnerabilidad que se presenta en la aplicaciones web la cual permite la ejecución de código java script o VBScript en los formularios de la misma.
[44] Script: Programa simple interpretado que facilita la interacción entre el usuario y el computador.
[45] Pagina web: http://www.microsoft.com/downloads/en/details.aspx?FamilyId=051ee83c-5ccf-48ed-8463-02f56a6bfc09&displaylang=en
[46] Pagina web: http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/Encoder.html
[47] Pagina web: http://wicket.apache.org/

Es necesario implementar validaciones correspondientes de datos de entrada en las tres capas correspondientes a la aplicación, se deben incluir controles en la capa de presentación de la aplicación como las restricciones, además se debe asegurar que los controles estén duplicados en el servidor (Servidor web, servidor de base de datos). Esta práctica minimizara las posibilidades de que un atacante pueda burlar los controles del lado del cliente mediante la manipulación de las cabeceras http[48].

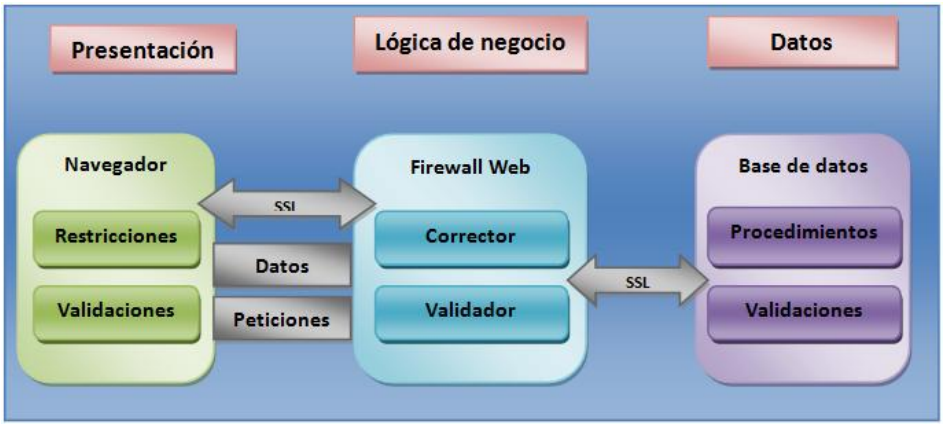
- **Definición de los componentes**

A continuación se describirá cada una de las funciones para cada componente dentro de la capa de negocio que permitirá la gestión de seguridad de los datos de entrada para la aplicación. (Ver figura 24)

- **Corrector.** El corrector deberá realizar las siguientes funciones:
  - Eliminación de los caracteres no deseados para la aplicación.
  - Eliminación de etiquetas HTML.
  - Eliminación de sentencias tipo script (Java script o VBScript).
- **Validador.** El validador deberá cumplir con las siguientes funciones:
  - Validación de los datos permitidos.
  - Omisión los datos no permitidos
  - Implementación controlada de los comandos del sistema.

[48] Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') [en línea]. CWE, 2010 [Consultado el 27 de Mayo de 2010]. Disponible en Internet: http://cwe.mitre.org/data/definitions/79.html

Figura 24. Esquema conceptual de componentes para la validación de datos de entrada



Fuente: Autor

9.4.4 Gestión de la configuración y manejo de errores

La gestión de la configuración de la aplicación se ejerce sobre la configuración y manejo de los servidores que la albergan, es por eso que las medidas presentadas a continuación harán énfasis principalmente en la configuración, luego se describirán aquellas medidas que se deben tener en cuenta para el manejo de errores generados por la aplicación.

- **Gestión de la configuración**

En resumen OWASP[49] y Microsoft[50] proponen para el manejo de la configuración de los servicios que soportan la aplicación web lo siguiente:

- Es importante deshabilitar todos los servicios innecesarios que se encuentren ejecutándose en el servidor que contiene a la aplicación web ya que consumen recursos y son una puerta abierta a los usuarios mal intencionados.

- No configure ningún servicio ni característica del servidor mientras no se

[49] Configuration. [en línea]. Owasp Foundation, 2010. [Consultado el 28 de Mayo de 2010]. Disponible en: http://www.owasp.org/index.php/Configuration
[50] Architecture and Design Review for Security
[En línea]. Microsoft, 2006. [Consultado el 29 de Mayo de 2010]. Disponible en: http://msdn.microsoft.com/en-us/library/ff648650.aspx#c05618429_009

necesite.

- Optimice al máximo la administración de los servicios para que sea lo más segura posible, revise los archivos de configuración de los servicios que se encuentran activos en el servidor, estos le ayudaran a personalizar las configuraciones.

- Se debe evitar confiar e instalar componentes opcionales ofrecidos por los programas o servicios.

- Se debe evitar utilizar productos de software que funcionen con cuentas por defecto.

- Verificar la existencia de puertas traseras o mecanismos de acceso especiales al servidor por parte de los servicios y programas que se encuentran instalados.

- La aplicación debe conectarse a la base de datos utilizando un usuario con el privilegio más bajo.

- La aplicación debe conectarse a la base de datos con diferentes credenciales para cada distinción de usuario (usuario de solo lectura, invitado, administrador entre otros) y permisos concedidos a las tablas de la base de datos para prevenir modificaciones y accesos no autorizados.

- Es aconsejable ubicar las bases de datos en otro servidor, que se encuentre asegurado con todos los parches y configurado con el último software de base de datos existente.

- Evite exponer la aplicación durante la etapa de desarrollo en la red, ya que no se encuentra totalmente implementada, un usuario mal intencionado podría llegar a manipular la aplicación aprovechando los posibles fallos de configuración.

Para compartir las configuraciones de seguridad existentes entre los servidores, es una buena práctica crear una imagen de un servidor asegurado y replicarla a los demás servidores. Mantener la configuración del servidor segura requiere vigilancia, debe estar seguro de que la responsabilidad por mantener la configuración del servidor actualizada es asignada a un individuo o equipo. El proceso de mantenimiento debería incluir:

- Monitorear las últimas vulnerabilidades de seguridad publicadas

- Aplicar los últimos parches de seguridad
- Escanear vulnerabilidades regulares a los servidores indicados.

➢ **Manejo de errores**

Algunos servidores (servidores web, de bases de datos entre otros) poseen un modo de depuración[51], el cual se recomienda habilitar únicamente cuando se esté desarrollando la aplicación, ya que la información mostrada podría ser aprovechada por un usuario mal intencionado para realizar futuros ataques[52].

Se deben capturar todos los errores provocados por la aplicación a través de sentencias o funciones dentro de cada lenguaje que estén diseñadas para tal objetivo.

Si el lenguaje de programación escogido posee una estructura de captura de excepciones[53] (estructura try{} catch()), puede ser utilizada como una solución para tal objetivo. Una vez capturados los errores, se debe mostrar al usuario un mensaje que sea claro, comprensible y que no muestre detalles técnicos del problema[54].

Cuando se muestren los errores producidos por la aplicación de manera controlada, se debe evitar mostrar información de contacto al usuario de la aplicación. Datos como correos electrónicos, nombres de personas o teléfonos pueden ser ideas grandiosas para un usuario mal intencionado quien podría llegar a generar un exploit[55] de ingeniería social[56].

[51]Depuración: Proceso que permite mostrar los errores que se generan en el software para una posterior corrección.
[52] Information Exposure through an Error Message [en línea]. CWE, 2010 [Consultado el 29 de Mayo de 2010]. Disponible en Internet: http://cwe.mitre.org/top25/#CWE-209
[53] Excepciones: Errores dentro de la ejecución normal de un software.
[54] Error Handling, Auditing and Logging [en línea]. OWASP, 2009 [Consultado el 29 de Mayo de 2010]. Disponible en Internet: http://www.owasp.org/index.php/Error_Handling,_Auditing_and_Logging#Error_Handling
[55] Exploit: Programa que se aprovecha de una vulnerabilidad.
[56] Ingeniería Social: Se describe a la ingeniería social como la técnica para poder engañar a las personas con el objetivo de obtener información confidencial.

➢ **Definición de los componentes**

A continuación se listarán cada una de las funciones para el componente perteneciente al manejo de errores. (Ver Figura 25)

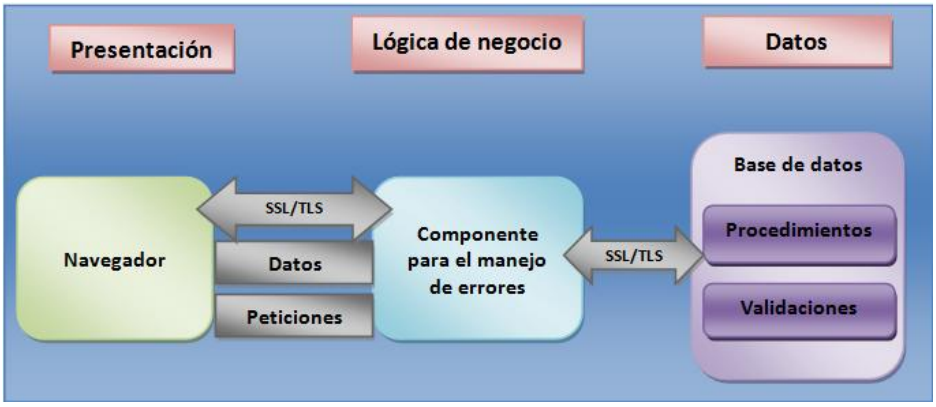
- **Componente para el manejo de errores.** El componente para el manejo de errores deberá cumplir las siguientes funciones:

  - Capturar los errores producidos por la base de datos.

  - Registrar los errores producidos por la aplicación.

  - Presentar los errores de una manera clara al usuario.

Figura 25. Esquema conceptual de componentes para manejo de errores en la aplicación.



Fuente: Autor.

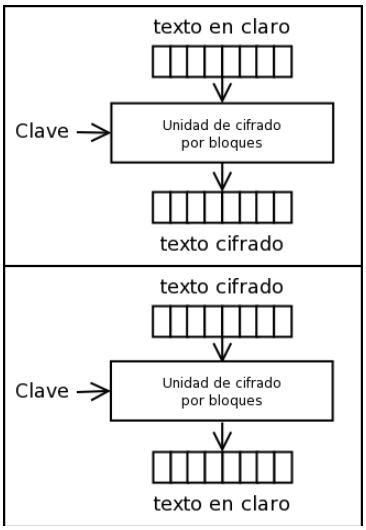
9.4.5 Gestión del almacenamiento seguro y transporte

➢ **Almacenamiento seguro**

Cuando se almacenen datos debe asegurarse de que se encuentren debidamente protegidos con mecanismos de cifrado simétricos. Un cifrado simétrico es aquel que utiliza una clave única para cifrar y descifrar datos e información (Ver Figura 26). Es conveniente utilizar una función o método de cifrado lo suficientemente fuerte ya que sobre ella recae todo el peso de la seguridad del algoritmo. A

continuación se mencionara en una tabla, algunos de los algoritmos de cifrado simétricos y sus principales características con respecto a su longitud de clave (Ver tabla 5):

Figura 26. Ilustración de un algoritmo de cifrado simétrico



Fuente: Cifrado por bloques. [en línea] Wikipedia, 2010 [Consultado el 03 de Junio de 2010]. Disponible en: http://es.wikipedia.org/wiki/Cifrado_por_bloques

Tabla 5. Clasificación de los algoritmos de cifrado simétrico más comunes según su longitud de llave.

Algoritmo	**Longitud de llave (bits)**
DES	64
AES	128/192/256
IDEA	128
RC5	variable
SAFER	64

Fuente: Autor.

Según Wu y Zhou[57] es importante que el algoritmo que cifra los datos se actualice conforme a las nuevas versiones para robustecer su longitud de llave, debido a

[57] WU, Xing-Hui. ZHOU, Yu-Ping. Analysis of Data Encryption Algorithm Based on WEB: Analyzes and comparation of the model. En: 2010 2nd International Conference on Computer Engineering

que el poder de procesamiento de las computadoras aumenta a través del tiempo. Se debe evitar guardar las llaves del algoritmo que descifran los datos en el mismo servidor donde se encuentra la información.

Es necesario que cuando genere las llaves de cifrado se haga en un equipo aislado de la red, una vez generadas las llaves es recomendable guardarlas con extremo recelo, además se recomienda no transmitir las llaves sobre canales inseguros. En la actualidad existen interfaces de cifrado existentes aprobadas por las comunidades de desarrollo de software ampliamente usadas para la protección de la información confidencial. Paquetes como JCE (Java Cryptography Extension)[58], el paquete BouncyCastle[59] o el paquete Encryptor de ESAPI[60] son conocidos por su popularidad y le ayudaran a implementar un almacenamiento de la información generada por la aplicación.

La implementación de algoritmos de cifrado por cuenta propia no es recomendable, se podrían presentar fallas de seguridad en su codificación. Es necesario utilizar los paquetes mencionados con anterioridad ya que estos han sido debidamente probados, aceptados y adoptados por grandes organizaciones a nivel mundial.

Según Meier[61], es innecesario, complejo y costoso implementar algoritmos de cifrado propios, ya que se verá reflejado en un pobre desempeño y seguridad del mismo. Puede destacarse a demás que la implementación propia de los algoritmos de cifrado es costosa en cuestiones de tiempo-dinero si se concibe que el equipo de desarrollo tardaría más tiempo en diseñar, desarrollar, implementar y probar un algoritmo que simplemente adoptar una librería o componente existente.

- **Transporte Seguro**

Es importante centrar la atención en la tecnología a utilizar en la aplicación web para garantizar un transporte seguro de la información entre el cliente y el servidor.

---

and Technology. 2010. p. 673-677

[58] Pagina web: http://java.sun.com/products/archive/jce/

[59] Pagina web: http://www.bouncycastle.org/

[60] Pagina web: http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/Encryptor.html

[61] MEIER, J.D, et al. Architecture and Design Review for Security: Cryptography [en línea]. Microsoft, 2006 [Consultado el 27 de Julio de 2010]. Disponible en Internet: http://msdn.microsoft.com/en-us/library/ff648650.aspx#c05618429_012

Una de las opciones más comunes es la adopción de TLS[62], protocolo comúnmente usado por las aplicaciones web y el comercio electrónico.

El principal beneficio de TLS es la apertura de canales seguros para la protección del flujo de datos que manipula la aplicación, TLS provee beneficios adicionales que están comúnmente relacionados:

- Garantiza la integridad y la prevención de la información que fluye a través de la aplicación.

- Garantiza únicamente el cifrado de los datos durante su transmisión, no garantiza el cifrado cuando lo datos se encuentran almacenados.

Según OWASP[63] es importante utilizar TLS en todas las páginas que tengan un formulario que permita enviar información al servidor. Los formularios de contacto y los formularios de acceso de usuarios son algunos de los cuales se podrían tener en cuenta al momento de proveer seguridad mientras su aplicación transporta información. Es necesario que todas las paginas que se muestren al usuario posterior a su autenticación se realicen implementando TLS.

- Se debe evitar mostrar páginas que presenten información confidencial mientras no usen TLS.

- Se debe evitar direccionar desde una página sin TLS a otra con TLS habilitado ya que inicialmente las credenciales de usuario no estarán protegidas.
- Debido a que TLS es el sucesor de SSL[64], es importante saber que no es recomendable utilizar versiones anteriores de SSL ya que se han demostrado falencias que podrían comprometer la confidencialidad de los datos mientras son transportados.

TLS funciona a través de certificados digitales. Los certificados digitales son documentos electrónicos los cuales a través de una entidad certificadora garantizan la vinculación entre un cliente y su clave pública. Una vez identificado el cliente, se prueba su identidad con el servidor. (Ver Figura 27)

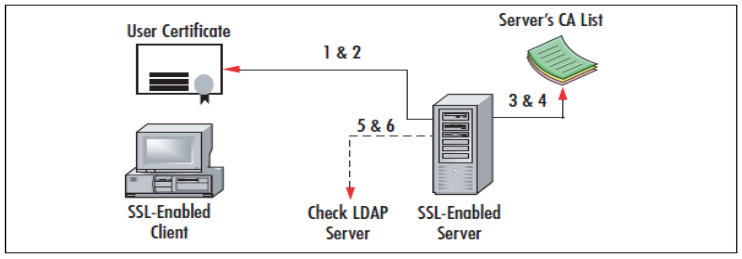
---

[62] TLS, Transport Layer Security o Seguridad en capa de transporte, es un protocolo usado entre la aplicación y la capa de transporte y permite el intercambio de información de manera segura entre cliente y servidor.

[63] Transport Layer Protection Cheat Sheet [en línea]. OWASP, 2010 [Consultado el 28 de Julio de 2010]. Disponible en Internet: http://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

[64] SSL, Secure Socket Layer. Antecesor de TLS.

Figura 27. Autenticación de un cliente mediante TLS y certificados



Fuente: CROSS, Michael. Developers guide to web application security. Syngress Publishing, 2007. p. 430.

- Al usuario de la aplicación no debe presentársele la advertencia de que el certificado fue firmado por una autoridad desconocida o no confiable. Es necesario que los certificados sean expedidos por una entidad certificadora quien estará autorizada para firmar y garantizar la confianza entre el cliente y el servidor.

- Se debe evitar firmar certificados sin la presencia de una entidad certificadora autorizada, no es una buena práctica ya que no garantizará la confiabilidad entre cliente y servidor.

- La fuerza del cifrado utilizado en una sesión TLS está determinada por el sistema de cifrado negociado entre el servidor y el navegador. Con el fin de garantizar buenos sistemas de cifrado, el servidor debe ser modificado para deshabilitar el uso de sistemas de cifrado débiles. Se recomienda utilizar tamaños de clave lo suficientemente grandes. En general, se deben especificar los sistemas de cifrado a utilizar.

➢ **Definición de los componentes**

A continuación se listaran cada una de las funciones para la gestión del transporte seguro. (Ver Figura 28)

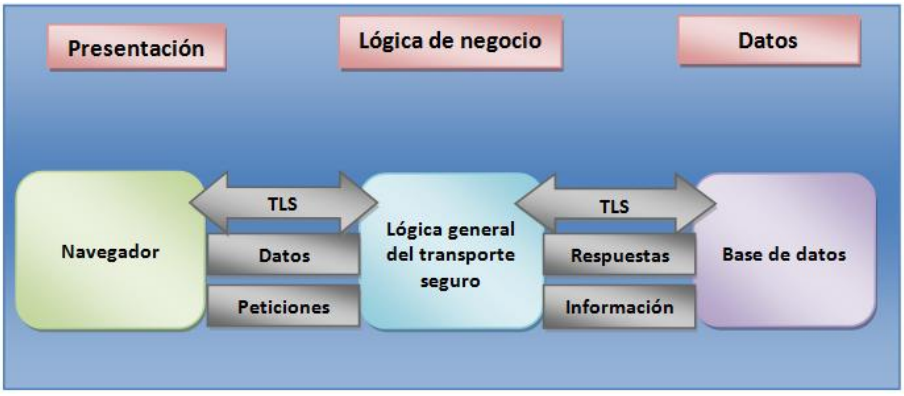
- **Lógica general del transporte seguro.** La lógica general para la gestión del transporte seguro deberá tener en cuenta las siguientes funciones:

  - Utilización de TLS mediante certificados para el transporte de la información.

  - Definición de un algoritmo de cifrado robusto.

- Garantizar el transporte seguro desde el navegador del usuario pasando por la aplicación web hasta el servidor de la base de datos.

Figura 28. Esquema conceptual de los componentes para gestión del transporte seguro.



Fuente: Autor.

9.4.6 Gestión de sesiones. El protocolo HTTP no posee estado, es decir que las preferencias de navegación y mensajes del protocolo no pueden ser recordados dentro de un periodo de tiempo determinado, es por eso que la responsabilidad de gestionar la concesión de sesiones entre el servidor y el cliente además de recordar las preferencias para cada usuario, pertenece exclusivamente a la aplicación.

Según Meier[65] es necesario tener en cuenta lo siguiente antes de considerar una solución a la gestión de las sesiones de la aplicación:

- Es necesario considerar un identificador de sesión que la aplicación utilice para administrar las sesiones de usuario, debe considerar también como se intercambiaran cuando el tiempo de sesión termine.

- Si realiza el seguimiento del estado de sesión con los identificadores de sesión. Se debe examinar si el identificador o cookie pasó por un canal cifrado como TLS. También deberá asegurarse de que el estado de la sesión sea almacenado de manera segura por la aplicación usando una cookie con

[65] MEIER, J.D, et al. Architecture and Design Review for Security: Session management [en línea]. Microsoft, 2006 [Consultado el 01 de Agosto de 2010]. Disponible en Internet: http://msdn.microsoft.com/en-us/library/ff648650.aspx#c05618429_012

identificador único que tenga caducidad.

Si utiliza autenticación de formularios, asegúrese de que su aplicación cifra las cookies de autenticación. Se recomienda utilizar TLS para llevar a cabo este proceso y evitar el robo de las cookies de inicio de sesión de los usuarios. Además de cifrar los identificadores durante su transporte, debe asegurarse que la aplicación almacene los datos de sesión en un área protegida contra el acceso no autorizado.

La aplicación debe establecer un tiempo prudente para un identificador de sesión. El tiempo del identificador de sesión deberá limitarse para mitigar la amenaza de un secuestro de sesión y robo o adivinación del identificador. Según OWASP[66] es recomendable asignar tiempos de sesión de 5 minutos para las aplicaciones de alto riesgo[67], 10 minutos para las aplicaciones de riesgo medio y 20 minutos para las aplicaciones de bajo riesgo.

Se debe implementar un algoritmo lo suficientemente aleatorio para que los identificadores de sesión no sean adivinados tan fácilmente por los usuarios mal intencionados. Debe tenerse en cuenta que el identificador de sesión deberá de estar vinculado a la dirección ip del cliente que se encuentra conectado.

Para evitar la falsificación de solicitudes y manipulación de las sesiones de usuario en las aplicaciones web, es necesario evitar almacenar datos o parámetros sensitivos dentro de las cookies. Algunas implementaciones utilizan los datos almacenados en las cookies para recordar preferencias de los usuarios o iniciar sesión de manera automática, por lo que muchas veces se almacenan identificadores de sesión o credenciales como nombres de usuarios y contraseñas de quienes utilizan la aplicación.

Implementar transacciones de múltiples pasos es una buena práctica contra la manipulación de solicitudes ya que permitirán al usuario navegar por varias páginas dentro de la misma aplicación web antes de finalizar una operación. Este tipo de prácticas generalmente es utilizada para operaciones de alto riesgo como las transacciones bancarias.

[66] Session Management: Architectural goals [En línea]. OWASP, 2010 [Consultado el 28 de Julio de 2010]. Disponible en Internet: http://www.owasp.org/index.php/Session_Management#Architectural_Goals

[67] Riesgo, es la probabilidad de que un evento ocurra, en este contexto consideramos al riesgo como la probabilidad de que una amenaza se aproveche de una vulnerabilidad informática.

Generar identificadores o tokens para cada solicitud que el cliente realice al servidor, los tokens deberán ser insertados dentro de los formularios HTML y enlaces relacionados con las operaciones de alto riesgo de la aplicación. Esta práctica forzará a la aplicación a verificar cada solicitud enviada por el usuario y disminuirá el riesgo de sufrir ataques de manipulación de parámetros. Es recomendable que tanto los identificadores de sesión de la aplicación como los identificadores de solicitud se envíen como campos ocultos al servidor.

La aplicación debe generar dos cookies para el control de la sesión de usuario. Una cookie se almacenará en el ordenador del cliente y otra cookie permanecerá como oculta mientras se mantenga activa la sesión. Cualquier solicitud que se realice a la aplicación web, deberá comparar ambas cookies; tanto la que se encuentre en el ordenador del cliente como la que se encuentre registrada como oculta en la aplicación. Los identificadores de ambas cookies serán comparados y deberán coincidir para evitar ataques de manipulación de sesiones y falsificación de solicitudes.

En la actualidad existen módulos para la prevención e implementación de soluciones contra los ataques destinados a vulnerar la sesión de los usuarios que utilizan las aplicaciones web. CSRFGuard[68] provee varios mecanismos para prevenir manipulación de sesiones, cuenta con una solución generadora de códigos Captcha, autenticación de sesiones mediante credenciales de usuario, y generación aleatoria de tokens o identificadores de sesión.

El proyecto ESAPI[69] ofrece una colección de funcionalidades llamada HTTPUtilities[70], la cual permitirá al equipo de desarrollo la implementación de la seguridad perteneciente a las solicitudes de usuarios, respuestas, sesiones, cookies y cabeceras de la aplicación web.

- **Definición de los componentes**

A continuación se listarán cada una de las funciones para la gestión de sesiones. (Ver Figura 29)

[68] Pagina web: http://www.owasp.org/index.php/Category:OWASP_CSRFGuard_Project
[69] ESAPI, Enterprise Security API. Es una librería para la codificación segura de aplicaciones web desarrollada por OWASP.
[70] Pagina web: http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/HTTPUtilities.html

- **Componente para el manejo de solicitudes.** El componente para el manejo seguro de las solicitudes deberá tener en cuenta las siguientes funciones:

  - Emplear transacciones de múltiples pasos para las operaciones críticas.

  - Generar identificadores o tokens para cada solicitud considerada crítica para la aplicación.

  - Identificar y controlar los valores de las cookies ocultas por la aplicación y las cookies almacenadas en el cliente.

  - Emplear un mecanismo para la finalización de transacciones críticas, por ejemplo la utilización de códigos CAPTCHA. CAPTCHA es un mecanismo que permite la diferenciación entre humanos y maquinas, es comúnmente utilizado en las aplicaciones web para la prevención de ataques automatizados contra los formularios.

- **Componente para el manejo de sesiones.** El componente para el manejo seguro de sesiones deberá tener en cuenta las siguientes funciones:

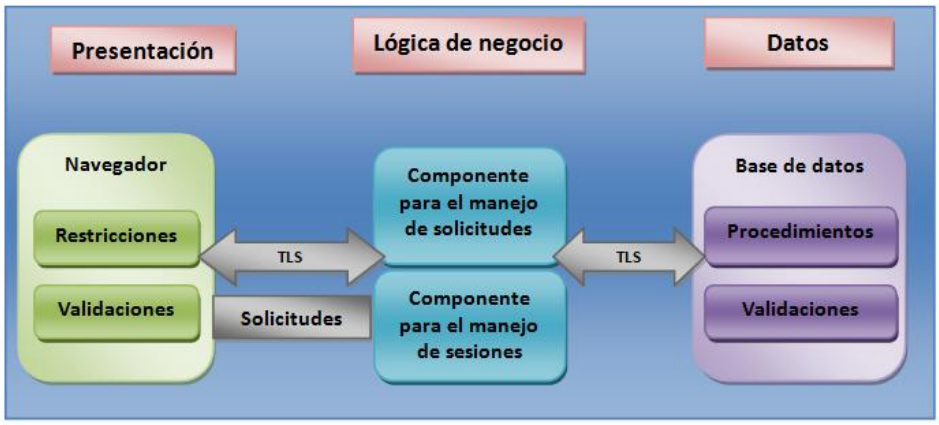
  - Generar los tokens o identificadores para cada sesión de usuario.

  - Realizar un seguimiento de estado a la sesión.

  - Asignar tiempos de vida a las sesiones de acuerdo a la importancia de las operaciones de la aplicación.

  - Cifrar las cookies para el control de las sesiones.

Figura 29. Esquema conceptual de los componentes para gestión de sesiones



Fuente: Autor.

10. CONCLUSIONES

- Con la investigación de las vulnerabilidades consideradas como críticas por los observatorios, comunidades y artículos consultados, se pudo determinar la importancia y el riesgo que representan para la información que manejan las aplicaciones web.

- El uso de la metodología OWASP para llevar a cabo las pruebas de penetración en el gestor metodológico Webgoat, permitió un análisis y entendimiento más profundo que contribuyó a la generación de recomendaciones y soluciones desde la arquitectura de la aplicación.

- La generación de recomendaciones basadas en las prácticas de los laboratorios realizados para entender y explotar las vulnerabilidades de las aplicaciones web permitió contemplar componentes definidos con funciones específicas para mitigar las vulnerabilidades web mencionadas.

- Las nuevas vulnerabilidades pueden ser contempladas por la metodología ya que la organización y clasificación de las mismas podrá realizarse mediante la matriz propuesta en la segunda fase de la metodología.

- Los entregables propuestos por la metodología permitirán organizar las ideas propuestas para la generación de los escenarios seguros. La metodología y los entregables pueden ser adaptados con facilidad a los requerimientos de seguridad de cualquier aplicación web.

- El uso y adopción de la metodología podría minimizar los costos de las soluciones correctivas de seguridad cuando las aplicaciones web se encuentren operativas, ya que el análisis profundo de las vulnerabilidades y la definición de componentes desde la fase de diseño de la aplicación ayudaran a mitigar los riesgos cuando sean contempladas por el arquitecto.

- El estudio de todas las vulnerabilidades que sufren las aplicaciones web es muy complejo, es necesario entonces analizar las vulnerabilidades y tendencias publicadas por los observatorios de seguridad informática y comunidades consolidadas a nivel mundial.

11. RECOMENDACIONES

- Si desea seguir la metodología en cualquier entorno, se recomienda realizar con anterioridad los análisis de requerimientos funcionales y no funcionales correspondientes al proyecto de software que desee asegurar.

- Aunque en este proyecto se buscó inicialmente generar una solución única para el control y mitigación de las vulnerabilidades en las aplicaciones web, fue muy difícil concebir una idea que agrupara toda la investigación realizada. Por eso se recomienda pensar siempre los detalles correspondientes a la seguridad de la aplicación y agruparlos en componentes que permitan definir lo que se pretende proteger.

- Para futuras adaptaciones o versiones de esta metodología, se sugiere realizar un estudio actualizado de las vulnerabilidades de seguridad existentes, se debe consultar en fuentes confiables como observatorios de seguridad, estos registran las nuevas amenazas y problemas de seguridad relacionadas con la informática.

12. BIBLIOGRAFIA

1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems. [en línea]. IEEE Computer Society, 2004. [Consultado el 24 de Julio de 2010]. Disponible en: http://standards.ieee.org/findstds/standard/1471-2000.html

Aplicaciones Web. [en línea]. Wikipedia. [Consultado el 17 de Abril de 2010]. Disponible en: http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web

Cleartext Transmission of Sensitive Information. [en línea]. CWE, 2010 [Consultado el 29 de Mayo de 2010]. Disponible en: http://cwe.mitre.org/data/definitions/319.html

Configuration, [en línea]. OWASP Foundation, 2010. [Consultado el 28 de Mayo de 2010]. Disponible en Internet: http://www.owasp.org/index.php/Configuration

CROSS, Michael. Developers guide to web application security. Rockland: Syngress Publishing, 2007. 513 p.

Cross-Site Request Forgery (CSRF). [en línea]. CWE, 2010 [Consultado el 29 de Mayo de 2010]. Disponible en: http://cwe.mitre.org/data/definitions/352.html

Error Handling: Auditing and Logging [en línea]. OWASP Foundation, 2009. [Consultado el 29 de Mayo de 2010]. Disponible en: http://www.owasp.org/index.php/Error_Handling,_Auditing_and_Logging#Error_Handling

Guide to Authentication: Architectural goals. [en línea]. OWASP Foundation. [Consultado el 23 de Mayo de 2010]. Disponible en: http://www.owasp.org/index.php/Guide_to_Authentication

Guide to Authorization: Principle of least privilege [en línea]. OWASP Foundation, 2009 [Consultado el 25 de mayo de 2010]. Disponible en: http://www.owasp.org/index.php/Guide_to_Authorization

HUAWEI, Zhao., RUIXIA, Liu. A Scheme to Improve Security of SSL. En: 2009 Pacific-Asia Conference on Circuits, Communications and System. 2009. p. 401-404.

Improper Access Control (Authorization): Potential mitigations. [en línea]. CWE, 2010. [Consultado el 25 de Mayo de 2010]. Disponible en: http://cwe.mitre.org/data/definitions/285.html

Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'). [en línea]. CWE, 2010. [Consultado el 27 de Mayo de 2010]. Disponible en: http://cwe.mitre.org/data/definitions/79.html

Information Exposure Through an Error Message. [en línea]. CWE, 2010 [Consultado el 29 de Mayo de 2010]. Disponible en: http://cwe.mitre.org/top25/#CWE-209

Insecure direct object reference: Protection. [en línea]. OWASP Foundation, 2010. [Consultado el 25 de Mayo de 2010]. Disponible en: http://www.owasp.org/index.php/Top_10_2007-Insecure_Direct_Object_Reference

Introducción a la informática: Definición de escenario. [en línea]. [Consultado 17 de Marzo del 2010]. Disponible en: http://dircompucv.ciens.ucv.ve/generador/sites/int-a-la-inf/archivos/Modelos-1.ppt

LIN, Xiaoli., ZAVARSKY, Pavol., RUHL, Ron., LINDSKOG, Dale. Threat Modeling for CSRF Attacks. En: 2009 International Conference on Computational Science and Engineering. 2009. p. 486-491.

MAGUIRE, John R., GILBERT MILLER, H. Web Application Security from Reactive to Proactive. En: IEEE Computer Society. 2010. p. 7-9.

MEIER, J.D. Web Application Security Engineering: Security engineering baseline activities. En: IEEE Security & Privacy. 2006. p. 16-24.

MEIER, J.D., MACKMAM, Alex., DUNNER, Michael., VASIREDDY, Srinath Ray.,

ESCAMILLA., y MURUKAN, Anandha. Architecture and Design Review for Security: Cryptography. [en línea]. Microsoft, 2006. [Consultado el 23 de mayo de 2010]. Disponible en: http://msdn.microsoft.com/en-us/library/ff648650.aspx#c05618429_012

--------.--------. Session management. [en línea]. Microsoft, 2006. [Consultado el 03 de Junio de 2010]. Disponible en: http://msdn.microsoft.com/en-us/library/ff648650.aspx#c05618429_012

--------.--------. Authentication. [en línea]. Microsoft, 2006. [Consultado el 23 de mayo de 2010]. Disponible en: http://msdn.microsoft.com/en-us/library/ff648650.aspx#c05618429_007

MEIER, J.D., MACKMAM, Alex. y WASTELL, Blaine. Cheat Sheet: Web application security frame. [en línea]. Microsoft, 2005. [Consultado el 22 de mayo de 2010]. Disponible en: http://msdn.microsoft.com/en-us/library/ff649461.aspx

MENDOZA GONZALEZ, Ricardo., MUÑOZ ARTEAGA, Jaime., VARGAS MARTIN, Miguel., ALVAREZ RODRIGUEZ, Francisco., y GONZALEZ CALLEROS, Juan. A Pattern Methodology to Specify Usable Security in Websites. En: 20th International Workshop on Database and Expert Systems Application.2009. p. 155-159.

NICHOLS, Elizabeth. A., PETERSON, Gunnar. A Metrics Framework to Drive Application Security Improvement. En: IEEE Security & Privacy. 2007. p. 88-91.

OWASP testing guide. 3 ed. OWASP Foundation, 2008. 349 p.

OWASP Top 10, [en línea]. OWASP Foundation. [Consultado 26 de Marzo de 2010]. Disponible en: http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf

Plaintext Storage in a Cookie. [en línea]. CWE, 2010 [Consultado el 29 de Mayo de 2010]. Disponible en: http://cwe.mitre.org/data/definitions/315.html

Plaintext Storage in a File or on Disk. [en línea]. CWE, 2010 [Consultado el 29 de Mayo de 2010]. Disponible en: http://cwe.mitre.org/data/definitions/313.html

Plaintext Storage in Executable.[en línea]. CWE, 2010 [Consultado el 29 de Mayo de 2010]. Disponible en: http://cwe.mitre.org/data/definitions/318.html

Plaintext Storage in GUI.[en línea]. CWE, 2010 [Consultado el 29 de Mayo de 2010]. Disponible en: http://cwe.mitre.org/data/definitions/317.html

Plaintext Storage in Memory.[en línea]. CWE, 2010 [Consultado el 29 de Mayo de 2010]. Disponible en: http://cwe.mitre.org/data/definitions/316.html

Plaintext Storage in the Registry.[en línea]. CWE, 2010 [Consultado el 29 de Mayo de 2010]. Disponible en: http://cwe.mitre.org/data/definitions/314.html

PRADO DUQUE, Leidy Johanna., SERRATE VALENCIA, Jorge Mario. Definición, Diagnostico y Adecuación de Políticas de Seguridad Lógica para la Página Web Comercial de Almacenes la 14 S.A Basado en los Requisitos Incluidos en la Norma PCI. Trabajo de grado Ingeniero Informático. Santiago de Cali: Universidad Autónoma de Occidente. Facultad de Ingeniería. Departamento de Ciencias de la Información, 2008. 118 p.

Reversible One-Way Hash.[en línea]. CWE, 2010 [Consultado el 29 de Mayo de 2010]. Disponible en: http://cwe.mitre.org/data/definitions/328.html

SCOTT, David., SHARP, Richard. Developing Secure Web Applications. En: IEEE Internet Computing. 2002. p.38-45.

Seguridad Informática. [en línea]. Wikipedia. [Consultado 11 de Abril de 2010]. Disponible en: http://es.wikipedia.org/wiki/Seguridad_inform%C3%A1tica

Session Management: Architectural goals. [en línea]. OWASP Foundation, 2010. [Consultado el 28 de Junio de 2010]. Disponible en: http://www.owasp.org/index.php/Session_Management#Architectural_Goals

STUTTARD, Dafydd., PINTO, Marcus. The Web Application Hackers Handbook: Core defense mechanisms. Wiley Publishing, 2008. p 28.

Transport Layer Protection Cheat Sheet [en línea]. OWASP Foundation, 2010 [Consultado el 28 de Junio de 2010]. Disponible en: http://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

TSAI, Dwen Ren., CHANG, Allen Y., LIU., Peichi. Optimum Tuning of Defense Settings for Common attacks on the Web Applications. En: IEEE. 2009. p. 89-94.

Url Redirection to Untrusted Site ('Open Redirect'): Potential mitigations. [en línea].Common Weakness Enumeration, 2010. [Consultado el 24 de mayo de 2010]. Disponible en: http://cwe.mitre.org/data/definitions/601.html

Weak Cryptography for Passwords.[en línea]. CWE, 2010 [Consultado el 29 de Mayo de 2010]. Disponible en: http://cwe.mitre.org/data/definitions/261.html

WU, Xing-Hui., ZHOU, Yu-Ping. Analysis of Data Encryption Algorithm Based on Web: Analizes and comparation of the model. En: 2010 2nd International Conference on Computer Engineering and Technology. 2010. p. 673-677.

ANEXOS

Anexo A. Listado de pruebas de penetración

Este anexo contiene el listado de las pruebas que se realizaron para el estudio de las vulnerabilidades por cada categoría definida por la metodología. Se encuentra en formato digital con el nombre “Anexo A. Listado de pruebas de penetración.pdf” dentro de la carpeta anexos, ubicada en el CD.

Anexo B. Laboratorios

Este anexo contiene los informes y laboratorios resueltos para cada una de las pruebas de presentación definidas en el Anexo A, se encuentran en formato digital dentro de una carpeta con el nombre “Anexo B. Laboratorios”, dentro de la carpeta anexos, ubicada en el CD.

A continuación se muestra un listado de los documentos incluidos en la carpeta del Anexo B:

- **Validación de datos de entrada**

  - Inyección de parámetros.
    - Documento “Add Data With Sql Injection.doc”.
    - Documento “Numeric Sql Injection.doc”.
    - Documento “Command Injection.doc”.
    - Documento “Database Backdoors.doc”.
    - Documento “Log Spoofing.doc”.
    - Documento “Modify Data with Sql Injection.doc”.
    - Documento “String Sql Injection.doc”.
    - Documento “Xpath Injection.doc”.
  - Vulnerabilidades Xss.
    - Documento “xss.doc”.

- **Autenticación**

  - Fallas de autenticación.
    - Documento “Basic Authentication.doc”.
    - Documento “Forgot Password.doc”.
    - Documento “Multilevel Login 1.doc”.
    - Documento “Multilevel Login 2.doc”.
    - Documento “Password Strength.doc”.

- **Autorización**

  - Referencia directa e insegura a un objeto.
    - Documento “Insecure Direct Object Reference.doc”.
  - Fallas de restricción de url.
    - Documento “Failed to Restrict Url Access.doc”
  - Direccionamientos no autorizados.
    - Documento “Unvalidate Redirect and Forwards.doc”.

- **Manejo de sesiones**

  - Vulnerabilidades de sesión.
    - Documento “Hijack a Session.doc”.
    - Documento “Spoof a Cookie.doc”.
  - Cross Site Request Forgery.
    - Prueba “Cross Site Request Forgery.doc”.
    - Prueba “Cross Site Request Forgery Bypass.doc”.
    - Prueba “Cross Site Request Forgery Token Bypass.doc”.

- **Manejo de la configuración**

  - Configuración insegura de la aplicación.
    - Prueba “Security Misconfiguration.doc”.

- **Criptografía**

  - Cifrado y almacenamiento inseguro.
    - Prueba “Insecure Criptographic Storage.doc”.
  - Protección insuficiente en la capa de transporte.
    - Prueba “Transporte.doc”

Anexo C. Gestor metodológico Webgoat

Este anexo contiene el gestor metodológico para el desarrollo de las pruebas listadas en el Anexo A, se encuentran en formato digital dentro de una carpeta con el nombre “Anexo C. Gestor metodológico Webgoat”, dentro de la carpeta anexos, ubicada en el CD.